Maximum-Matching based Maintenance of Structural Controllability

Shuo Zhang

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

2019

Maximum-Matching based Maintenance of Structural Controllability

Department of Mathematics
Royal Holloway, University of London

Young men should prove theorems, old men should write books.

(G. H. Hardy)

Declaration of Authorship

I, Shuo Zhang, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed: **Shuo Zhang**

(Shuo Zhang)

Date:03/03/2020

Abstract

Controllability [63] as a critical property of the dynamical system, validates proposed services, functions and products in human society. To effectively derive minimum-input control into a continuous time and linear time-invariant (CT-LTI) system, maximum-matching based method is raised to derive minimum-input structural controllability [72]. This is because a structurally controllable system is controllable with high probability. As another result, controllability of complex networks with CT-LTI dynamics is further studied, where any single network vertex directly forced by inputs, is called the driver node. Obviously, maliciously attacker can hijack or damage the control through deriving or sabotaging structural control into a same system. Therefore, by the maximum-matching based method, this thesis is highly motivated to maintain minimum-input structural controllability in two parts. To defend against very limited and severe modification, the first part focuses on the efficient structural-control recovery. The second part is about efficient network analysis on all single nodes and edges that are vulnerable to removal in maintaining current minimum-input structural controllability.

In the first part, the first research question focuses on efficient recovery after adding a single vertex into an initially minimum-input structurally controllable network, whose related research paper can refer to [127]. The next one is about minimum-input structural-control recovery after removing a single system component from an initially minimum-input structurally controllable system, whose research paper can refer to [129]. By contrast, the third question aims at recovery of structural controllability with a fixed number of given inputs after severe modification on system components, whose research paper can refer to [126].

In the second part, because removal of single edges or vertices can dramatically increase the minimum number of inputs to recover structural control into residual system, another three research questions are defined and solved. The fourth question is about more efficiently classifying all edges of an initially minimum-input structurally controllable network into well-defined critical, redundant and ordinary categories, whose research papers can refer to [130, 128]. After the edge-based analysis, given a digraph with CT-LTI dynamics, the fifth research question not only finds single vertices of all minimum sets of driver nodes, but also classifies them by importance of any single driver node in maintaining a minimum set of inputs, whose research paper can refer to [125]. The last research question addressed by this thesis proposes to efficiently classify all single vertices of an initially minimum-input structurally controllable network, according to the importance of any single vertex in maintaining the current minimum set of inputs.

Generally speaking, compared with related works, those six questions are more efficiently solved with less assumptions in lower worst-case complexity.

Acknowledgments

I would like to sincerely thank my supervisor, Professor. Stephen D. Wolthusen, for his support and guidance in my entire PhD study. It was a great challenge for me, who used to study law, to do research mainly based on mathematics. He helped me with the transition by precisely defining research questions and giving continuous feedback on my work. Without his supervision, this thesis would not have been possible. I would also like to thank my advisor, Professor Chris Mitchell, who accompanied my development via the annual reviews, and recommended math books for me.

My heartfelt thanks to my parents, for persuading me to take up a PhD program, for their financial support over years, and for their thoughtful kindness and immeasurable care. Simultaneously, I must also thank my grandmother, who is the mother of my father, and my late grandfather, who is the father of my mother, for their unconditional love and encouragement. Most impressively, both of them selflessly gave their savings to me, while they had no regular income for years. Be loved by these family members via various ways, I therefore had courage and confidence to overcome difficulties in both study and life in the United Kingdom. Thousands of words are not enough to express my thankfulness for them.

Contents

I Overture **1**

1 Introduction **3**
1.1 Overview . . . 3
1.2 General Motivation . . . 3
1.3 Research Questions . . . 10
1.4 Contributions . . . 15
1.5 Layout . . . 15

2 Background **19**
2.1 Overview . . . 19
2.2 Controllability of CT-LTI systems . . . 19
2.3 Structural Controllability . . . 21
2.4 Derive Structural Controllability . . . 26
2.5 Strongly Structural Controllability . . . 33
2.6 Controllability of Complex Network . . . 37

3 Related Works **39**
3.1 Overview . . . 39
3.2 Structural-Controllability Recovery . . . 39
3.3 Robustness of Network Structural Controllability . . . 42
3.4 Network Analysis for Structural Controllability . . . 45

II Efficient Structural-Controllability Recovery **51**

4 Iterative Recovery of Structural Control by the Maximum Matching **53**
4.1 Overview . . . 53
4.2 Problem Formulation . . . 54
4.3 Solution . . . 55
4.4 Summary . . . 61

5 Structural-Control Recovery for Resilient Control Systems **63**
5.1 Overview . . . 63
5.2 Problem Formulation . . . 64
5.3 Solution . . . 65
5.4 Summary . . . 75

6 Structural-Control Recovery via the Minimum-edge Addition **77**
6.1 Overview . . . 77
6.2 Problem Formulation . . . 78

6.3 Disjoint Cacti Construction . . . 79
6.4 Summary . . . 88

III Efficient Network Analysis to Maintain Structural Controllability 89

7 Security-Aware Edge Analysis for Structural Controllability 91
7.1 Overview . . . 91
7.2 Preliminaries & Problem Formulation . . . 92
7.3 Identification of Arcs of Maximum Matchings . . . 94
7.4 Summary . . . 101

8 Driver-Node based Analysis for Structural Controllability 103
8.1 Overview . . . 103
8.2 Preliminaries & Problem Formulation . . . 104
8.3 Solution . . . 106
8.4 Summary . . . 114

9 Identify Vulnerable Nodes for Network Structural Control 115
9.1 Overview . . . 115
9.2 Problem Formulation . . . 116
9.3 Nodal Categories . . . 117
9.4 A Single-Vertex Classification . . . 121
9.5 Entire Nodal Classification . . . 124
9.6 Summary . . . 133

IV Epilogue 135

10 Conclusion & Future Work 137
10.1 Thesis Summary . . . 137
10.2 The Future Work . . . 139

Bibliography 141

List of Definitions

2.1 Controllability [99] . . . 19
2.2 Controllability of a DT-LTI System [19] . . . 20
2.3 Structural Controllability [72] . . . 22
2.4 Digraph G(A, B) . . . 22
2.5 Stem & Bud[72] . . . 22
2.6 Dilation of a Digraph[72] . . . 22
2.7 Inaccessibility [72] . . . 22
2.8 Cactus[72] . . . 23
2.9 Maximum Matching of a Digraph[35, 14] . . . 26
2.10 Maximum Matching of a Bipartite or Undirected Graph [35, 14] . . . 26
2.11 Perfect Matching[35, 14] . . . 26
2.12 Augmenting Path [60] . . . 26
2.13 A Dominating Set of a Graph [55] . . . 29
2.14 Domination Number [55] . . . 29
2.15 Observation Rules of undirected graphs [67] . . . 30
2.16 A Power Dominating Set of an Undirected Graph [59] . . . 30
2.17 Observation Rules of Digraphs . . . 30
2.18 A Power Dominating Set of a Digraph . . . 30
2.19 Strongly Structural Controllability . . . 34
2.20 Bipartite Graph B_A . . . 34
2.21 *Constrained t-matching* [54] . . . 34
2.22 Self-less Matching . . . 35
2.23 Matrix A_X . . . 35
2.24 Bipartite Graph B_{A_X} . . . 35
4.1 Input Network of chapter 4 . . . 54
4.2 . . . 54
5.1 Input Network of chapter 5 . . . 64
5.2 . . . 68
6.1 Input Network of chapter 6 . . . 78
6.2 S_{cc} . . . 82
7.1 Input Network of chapter 7 . . . 92
7.2 . . . 94
7.3 Alternating-Cycle Matchig . . . 94
7.4 Alternating-Path Matching . . . 94
8.1 Input Network of chapter 8 . . . 104
8.2 . . . 105
8.3 Extra-unmatched Node . . . 107
8.4 Ordinary & Spare Node . . . 112
9.1 Input Network of Chapter 9 . . . 116

9.2 117
9.3 Pre-augmenting path 118
9.4 Nodal Categories 121
9.5 $G = (V_G, E_G)$ 128

List of Figures

2.1 Cacti 23
2.2 A digraph of the system of equation (2.1) by definition 2.4. 24
2.3 Digraphs mapped by four simple CT-LTI systems. 24
2.4 Examples of Augmenting Paths 27
2.5 A Maximum Matching & A Cacti 28
2.6 A Minimum Dominating Set of an Undirected Graph 30
2.7 A Power Dominating Set of a Digraph 31
2.8 Relationships among Basic Control Concepts 34
2.9 Examples of *constrained t-matchings* 35
2.10 Examples of bipartite graphs for strongly s-controllable systems 36

4.1 An example of bijections of definition 4.2. 55
4.2 57

5.1 66
5.2 66
5.3 67
5.4 68

6.1 85

7.1 An *alternating-path* matching and an *alternating-cycle matching* 95
7.2 96
7.3 98

8.1 105
8.2 108

9.1 118
9.2 Pre-augmenting Paths. 119
9.3 An example of definition 9.5. 129

List of Theorems

2.1 Structural Controllability Theorem [72, 79] . . . 23
2.6 Strongly S-Controllable System Theorem . . . 36
2.8 Minimum Input Theorem [73] . . . 38
4.1 . . . 56
5.4 . . . 68
5.5 . . . 69
6.8 Edge-addition Scenario . . . 83
6.9 . . . 84
7.1 . . . 93
7.3 . . . 94
7.4 . . . 95
7.5 . . . 96
8.3 . . . 107
8.6 . . . 111
9.2 . . . 118
9.3 . . . 119
9.4 . . . 120

List of Lemmas

5.1 65
5.3 67
6.1 79
6.5 82
8.1 105
8.2 106
8.7 112
9.1 117
9.6 121
9.7 122
9.8 122

List of Corollaries

2.2 24
2.3 26
2.4 27
2.5 29
2.7 36
4.2 57
4.3 57
4.4 Time complexity of algorithm 4.1 59
4.5 Time complexity of algorithm 4.3 61
5.2 66
5.6 Time complexity of algorithm 5.1 71
5.7 Time complexity of algorithm 5.2 72
5.8 Time complexity of algorithm 5.3 74
5.9 Time complexity of solving problem 5.2 74
6.2 80
6.3 80
6.4 Time complexity of algorithm 6.1 81
6.6 83
6.7 Time complexity of algorithm 6.2 83
6.10 85
6.11 Time complexity of algorithm 6.3 87
7.2 93
7.6 Time complexity of algorithm 7.1 97
7.7 Time complexity of algorithm 7.2 99
7.8 Time complexity of algorithm 7.3 99
8.4 109
8.5 Time complexity of identifying all single driver nodes 110
8.8 112
8.9 113
8.10 114
9.5 120
9.9 Complexity of algorithm 9.2 125
9.10 Complexity of algorithm 9.3 126
9.11 Complexity of algorithm 9.4 127
9.12 Complexity of algorithm 9.5 130
9.13 Complexity of algorithm 9.6 132
9.14 Complexity of solving problem 9.2 132

List of Algorithms

2.1 Identify V_2 and E_2 by a maximum matching of $G(\mathbf{A})$. 28
2.2 Identify a power dominating set of $G(\mathbf{A})$. 31
2.3 Identify V_2 and E_2 by a power dominating set of $G(\mathbf{A})$. 32

4.1 Identify an augmenting path incident to u^+ related to M_B. 58
4.2 Identify a Maximum Matching of $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$. 60
4.3 Identify a maximum matching of digraph $(V \cup \{u\}, E \cup E_u)$. 60

5.1 Find a maximum matching with minimum number of edges incident to u^-. 70
5.2 Derive a maximum matching MM of D. 72
5.3 Derive a matching $M_{(v_g, v_f)}$. 73
5.4 Derive a maximum matching of $D \setminus \{u\}$. 74

6.1 The first edge-addition step. 81
6.2 The second edge-addition step. 83
6.3 Construct a digraph spanned by disjoint cacti 86

7.1 Produce a Digraph $D_0 = (V_0, E_0)$. 97
7.2 Identify arcs of directed paths and cycles. 98
7.3 Identify arcs contained by maximum matchings of D. 100

8.1 Find all existing unmatched nodes of V_B^- 110
8.2 Classify nodes of S_{n_1} 113

9.1 Overview of confirming category of each node of D 124
9.2 Find all leading vertices of V_B 125
9.3 Find terminals of pre-augmenting paths 126
9.4 Confirm value of $d \in \{0, -1\}$ 127
9.5 Set a label on each node of G 129
9.6 Confirm value of $d \in \{0, +1\}$ for all nodes of S_3 131

Part I

Overture

Chapter 1

Introduction

1.1 Overview

This thesis would be systematically introduced in this chapter. In detail, general reasons for determining our research areas and objectives are shown, in the first place. Then, dedicated motivation for raising research questions and those questions themselves are illustrated precisely. Besides, our relevant publications involved into solving those research questions are also listed in this chapter. At the end of this chapter, the layout of this thesis is given to present what each following chapter does for the purpose of solving every research question, where related contributions are also clarified as well.

1.2 General Motivation

In this section, some fundamental concepts are introduced, which are also research areas of this thesis. In the following, dynamical system, controllability, structural controllability, controllability of control systems and complex networks, would be introduced in detail. Next, with those concepts, objectives of this thesis are indicated in section 1.2.6. Furthermore, in section 1.3, the dedicated motivation for raising each research question is thus summarized, and each research question is closely corresponding to a dedicated motivation. For objectives and research questions, they are primarily about efficient structural-controllability recovery against the very limited and severe system-component modification in one aspect. Additionally, it is also concentrated to efficiently identify vulnerable single vertices and edges to the removal in terms of maintaining network structural controllability with a minimum set of inputs.

1.2.1 Dynamical System

State of a given system is regarded as an abstract quantity. Intuitively speaking, state is the minimum amount of information about the past history of a given system, and it also suffices to predict the effect of past inputs on the future. By contract, the input of a given system is considered as the force that acts on state of the system, while the directly observed state of the system is the output. With the state, input and output, a dynamical system as a mathematical structure is defined via following axioms [65]:

1. There are a state set Σ, input set Ω, output set Φ, and a time set Θ. In Θ, $\{\Sigma$, Ω,$\Phi\}$ are defined.

2. For any initial time $t_0 \in \Theta$, any initial state $x_0 \in \Sigma$ is defined, for any input $u \in \Omega$ defined at a moment $t \geq t_0$, the future state x_t of the system is determined by a transition function $\varphi : (\Omega \times \Theta) \times (\Theta \times \Sigma) \to \Sigma$, which is represented as: $\varphi_u(t; t_0, x_0) = x_t$ if and only if $t \geq t_0$. Moreover, for any $t_0 \leq t_1 \leq t_2$ in Θ, any state $x_0 \in \Sigma$, and any fixed $u \in \Omega$ defined over $[t_0, t_1] \subseteq \Theta$, the relations below hold:

   a) $\varphi_u(t_0; t_0, x_0) = x_0$;

   b) $\varphi_u(t_2; t_0, x_0) = \varphi_u\big(t_2; t_1, \varphi_u(t_1; t_0, x_0)\big)$.

   c) The system must be causality. For example, if the input $u, v \in \Omega$ and $u \equiv v$ defined on $t \in [t_0, t_1]$ and $[t_0, t_1] \subseteq \Theta$. Then, $\varphi_u(t; t_0, x_0) \equiv \varphi_v(t; t_0, x_0)$.

3. Each output of the system is a function $\psi : \Theta \times \Sigma \to \Phi$.

A system is said to be controllable within a time interval $[t_0, t_f] \subseteq \Theta$, for any initial state $x_0 \in \Sigma$ defined at t_0, and a proposed state $x_f \in \Sigma$ defined at time t_f, there must be an input set $\{u_t | t \in [t_0, t_f]\}$ that can steers x_0 to x_f [99]. In terminology, a controllable system is said to be equipped with the controllability, and the state of this system can be transferred from any given initial state to a proposed one by properly applying inputs within limited time [63]. With various mathematical models [58, 65] of dynamical systems, controllability as one of system properties has been studied through different areas, such as control systems [63], complex networks [73, 75, 110], autonomic computing [66, 107], and so on.

For example, in the area of control systems, in 1960s, Kalman [63, 65] originally raised an algebraic condition to model, design and analyse a continuous-time and linear-time invariant (CT-LTI) system by an ordinary differential equation, which is also called the state equation [65]:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{1.1}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the state matrix, and its every non-zero entry illustrates which system components interact with each other, and the value represents the strength of those interactions; here, any component of a system could be a physical variable, or a physical entity; $x(t) \in \mathbb{R}^n$, and $x(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T$ is the state vector, and it captures the state of each system component at time t; $u(t) \in \mathbb{R}^m$, $u(t) = (u_1(t), u_2(t), \ldots, u_m(t))^T (m \leq n)$, is the input vector, and holds external m inputs at time t; $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the input matrix, and its non-zero entries show which inputs force or drive which system components directly.

1.2.2 Controllability

In modern control theory [63], to understand if a dynamical system is equipped with the controllability or not, as mentioned in previous, it depends on whether the state of this system can be transferred from any given initial state to a proposed one within limited time by properly applying external inputs. In this thesis, controllability of continuous-time and linear time-invariant (CT-LTI) systems is only concentrated for following reasons.

For various dynamical systems studied over years, firstly, this is because the CT-LTI system is mostly studied, which is modeled by equation (1.1). When the system described by equation (1.1) is equipped with controllability, if and only if, the matrix $\mathbf{C} \in \mathbb{R}^{n \times nm}$, where $\mathbf{C} = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}]$, has full rank. Particularly, for this matrix $\mathbf{C}$, $\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}$ are n different submatrices of it. Clearly, to use this condition in practice, exact values of non-zero entries of matrix $\mathbf{A}$ and $\mathbf{B}$ of equation (1.1) must be known. For more details about this rank condition, please see section 2.2 of chapter 2.

The second reason is that the first step in any control challenge is said to establish the controllability of the locally linearized system [106]. Meanwhile, the third reason is that the condition of a completely controllable discrete time linear time invariant (DT-LTI) system is the same as that of a CT-LTI system [19], which would be clarified in section 2.2.1 of chapter 2. Additionally, the last reason is that many comprehensive, rigorous and detailed algebraic and graph-theoretical methods and concepts are created in order to derive controllability of CT-LTI systems, such as structural controllability [72], and strong structural controllability [80]. By now, these two control concepts have been widely used to study controllability in the perspective of graph-theoretical manners, which would be systematically introduced in section 2.3 and 2.5 of chapter 2.

1.2.3 Structural Controllability

One of very useful concepts to study controllability of CT-LTI system is structural controllability. A structurally controllable CT-LTI system means there must be at least one completely controllable system that shares the same structure of their both input and state matrices, so that it is very likely that a structurally controllable system is completely controllable. Due to this fact, given equation (1.1), acquiring controllability of CT-LTI system can be done through structural controllability without concerning those exact values of non-zero entries of $\mathbf{A}$ and $\mathbf{B}$, and calculating rank of $\mathbf{C}$.

Chronologically, in 1970s, by the graph interpretation of $\mathbf{A}$ and $\mathbf{B}$ of equation (1.1), Lin *et al.* [72] raised the structural controllability of a CT-LTI system to acquire its controllability. The graph interpretation of a given CT-LTI system is a digraph, whose each arc represents an unique non-zero entry of either $\mathbf{A}$ or $\mathbf{B}$, and it is defined by definition 2.4 of section 2.3 of chapter 2. Based on structural controllability, controllability of the given system can be thus effectively obtained with high probability in polynomial time. Meanwhile, by contrast, still based on the same graph interpretation, Mayeda and Yamada [80] defined strongly structural controllability of a CT-LTI system for the same purpose. They defined that once a system is equipped with strongly structural controllability, this system is always completely controllable without concerning values of non-zero entries of both input and state matrices.

Obviously, based on the rank condition mentioned in section 1.2.2, verifying a completely controllable CT-LTI system with a set of inputs might be computationally massive, let along confirming a minimum set of inputs [72] [31]. Fortunately, in past two decade, two graphic-theoretical methods are created to drive structurally controllable CT-LTI system with a minimum set of inputs. As a result of using that two methods, a completely controllable system with a minimum set of inputs can

be obtained with high probability. On the one hand, Liu *et al.* [73] achieved this objective by identifying a maximum matching [60] of that digraph, and the worst-case execution time is polynomial. Alternatively, another graph-theoretical method is to identify a power dominating set [59] of a same digraph [12, 6]. Nevertheless, identifying a minimum power dominating set is a NP-complete problem [59] in general. Even though, in reality, for different kinds of graphs, a minimum power dominating set can be identified with various time complexity [12]. Above all, given the matrix **A** of equation (1.1), problems about identifying an input matrix to construct a completely controllable CT-LTI system, can be solved by those graph-theoretic methods for constructing a structurally controllable CT-LTI system. For those algebraic and graph-theoretical methods to derive controllability, they are shown in section 2.3, 2.4 of chapter 2.

Due to the time-complexity efficiency of maximum matching based method that provides structural controllability with a minimum set of inputs, this thesis only applies this method to not only derive structural controllability with a minimum set of inputs, but also solve all research questions. Those research questions are primarily about recovering structural controllability, and identifying single network vertices and edges that are vulnerable to the removal in maintaining structural control with a minimum set of inputs.

1.2.4 Controllability of Control Systems

The control system integrates computing and communication capacities with monitoring and control of entities in the physical world [29, 27]. It can effectively sense, compute, communicate and control physical systems automatically [82] [69], which are also called process control systems, cyber-physical systems, distributed control system and supervisory control and data acquisition (SCADA) systems [26] in different applications [28]. For example, for oil industry, in a same and large area, a control system could be used to monitor and guide oil flow within the complex and large-scale oil pipelines. It which contains thousands or even tens of thousands of sensors, and actuators. Generally, such control systems are called SCADA. By contrast, for monitor and optimize interregional railway service, multiple control systems are arranged hierarchically. In each local railway network, a local control system is arranged. Also, those local control systems are interconnected to compose a larger control system, and such a larger system is called distributed control system. The physical process [108] or physical system [29] contained by a control system produces the output, and is the system proposed to be controllable or equipped with controllability via inputs. Due to the output of a physical system, the desired service or function is obtained eventually. Also, the physical system can be only represented by the matrix **A** of equation (1.1). For instance, given a oil pipeline, it is the physical system of the control system, and oil flow speed, pressure and other aspects is composed of the physical process of related control system. Given a control system, a physical system can be intuitively seen as the "system" of it, and a controllable physical system is equivalent with being equipped with controllability. Literally, controllability of a physical system of a control system is referred to the controllability of this control system in this thesis. To control the physical system, a control system should also contain other essential parts, including actuators, sensors, controllers and communication infrastructures to effectively measure, assess

and adjust outputs of the physical system, respectively. Control into the physical system can be acquired automatically or manually, and a control system can operate within an open loop, a close loop, or a human mode. For open-loop control systems, the physical system is forced by pre-defined settings, while the closed-loop one uses outputs to dynamically force the physical system, so that desired outputs can be produced at a moment. In the human mode, physical system is completely forced by humans.

Due to diverse capabilities and environmental coupling, control systems are pervasively used in controlling critical infrastructures [83], such as electrical, water, oil and natural gas, and discrete manufacturing [108]. Nevertheless, once the critical infrastructure is out of control for some reasons, it may lead unavailability of purposed products and services. If such unavailablity exists in a large region for a significant length of time, it would cause serious economic impacts or life loss [76]. Further, with the interdependenties among critical infrastructures [98], a single uncontrollable infrastructure may cause severe cascading and escalating failures. Particuarly, availability of the control into a physical system should be only dedicated for legal users of the control system. This is becausue control system might be operated based on the existing control into the physical system to meet malicious purposes, such as the case called the Maroochy Water Breach [105], and the Stuxnet attack on the Iranian uranium-processing equipment [36].

1.2.4.1 Attacks on Control Systems

In recent years, more and more research pays attention to securing control systems [29, 83, 96]. Here, Cardenas *et al.* [29] summarized few kinds of representative attacks, such as deception attacks, where adversary sends false information to controllers or sensors; Dos attack, where attacker prevents the controller or physical system from receiving sensor measurement or actuator signal; and physical attacks, where the physical system and other devices are directly damaged. In [96], except for physical attacks, given a cooling control system, other two kinds of attacks are simulated to observe their harmfulness. For the response to such attacks, state of the physical system would remain for an extra period of time. By comparison, once physical components are damaged, the control system would completely lose its control into the physical system.

Obviously, in addition to IT-based methods about securing control system, it is still desirable to guarantee the availability of control into a physical system. This is because control into physical system is necessarily determined by interactions among components of the physical system and inputs of a same control system, which might out of the range of IT security. As for an effective method, structural controllability of the given physical system is helpful to design and analyse interactions among physical system components and inputs.

In summary, it is indispensable to not only maintain the controllability of control systems continuously, but also recover the controllability of the system after some changes as soon as possible. And this thesis only concentrates on the recovery after attacks or failures on the physical system.

1.2.5 Controllability of Complex Networks

Control of complex networks is also an important research area [75, 73, 62, 37], and this thesis would also concentrate on controlling complex networks with CT-LTI dynamics according to structural controllability. This is because complex networks have been studied over decades [4, 87, 15, 119]. Besides, controlling a complex network is another research area in recent years, which enriches the understanding of complex networks through control theory.

Here, a CT-LTI dynamical network with inputs is equivalent to the physical system of a control system, and they can be also modelled by equation (1.1) as well, where the system component corresponds to a network vertex. In the past decade, based on structural controllability, Lin's graphic-interpretation of a given CT-LTI dynamical system, and Liu's maximum-matching based method, are both used to effectively derive the (structural) control into CT-LTI dynamical complex networks, such as interbank networks [46], and epidemic networks [103]. Particularly, in terms of statistical physics, where types of a given graph is strictly argued, and the given directed network is quite large in the number of both nodes and edges. With the background of statistical physics, based on Liu's scenario, the derived structurally controllable network with a minimum set of inputs is directly deemed as a completely controllable system. Nevertheless, it is true that a structurally controllable network is not completely controllable. An example is shown in section 2.3.1 of next chapter. In [117], the structurally controllable network that is not completely controllable, is further modified to obtain a completely controllable network, such as adding extra inputs. This finally resulting network is said to be physically controllable, or equipped with the physical controllability in comparison of structural controllability.

In this thesis, the aspect of statistical physics in solving research questions is not considered. The graph is assumed to be the large digraph, and the exact types of the assumed digraph is also not strictly constrained.

1.2.5.1 Attacks on Network Controllability

Simultaneously, robustness of network (structural) controllability with a minimum set of inputs against attacks [94] has been studied. Attacks on network controllability could be classified into single vertex removal, single edge removal, and cascading vertex or edge failures, respectively, which are specifically illustrated in section 3.3 of chapter 3. Nonetheless, with those graph-theoretical methods to derive structural controllability, malicious attackers can also use them to effectively analyse and damage controllability of CT-LTI networks by analysing vulnerable single nodes and edges. Therefore, in order to effectively recover and defend controllability against malicious attacks and failures, it is essential that we should further study those graph-theoretical methods in terms of recovering and analysing structural controllability with a minimum set of inputs.

1.2.6 Objectives

From the above, it have been known that structural controllability is useful to study and solve related problems about controllability of CT-LTI systems. Objectives of

this thesis are thus based on structural controllability.

Firstly, according to the maximum-matching based method, given a CT-LTI system, one objective of this thesis is to efficiently recover its structural controllability after either a very limited or severe component modification, rather than recomputing a maximum matching of the residual system. Also, the inputs for recovery after severe modification are fixed. Reasons of determining this objective is shown below:

Generally speaking, recovery of structural controllability attracts increasing attention in recent years, and people implement it by applying those two previously mentioned graph-theoretical methods [7, 48, 11], where computational efficiency for entire process of recovery is always considered.

Even though, one reason is that those existing works did not discuss the constraint of inputs that are used to recover structural control. Without discussing such constrains, the simply identified input matrix by those methods might not be used to recover structural conrollability of the residual system. This is because the number of simply calculated inputs is more than that of inputs in reality.

Another reason is that the amount of the change on the initial system also seems to be neglected by related works. Possibly, changes on the system component can be either very limited, such as adding or removing one single system component; or very severe, such as cascading failures on a large number of system components. Furthermore, those very-limited change might periodically emerge, and recovery of structural control is required per modifying a system. For example, in study of robustness of network controllability against the single node and edge removal [115] [100] [88], after a single node or edge removal, the minimum number of inputs to structurally control the latest network is calculated. Meanwhile, in terms of optimizing the minimum number of inputs [78] [118], the effect of optimization is reflected by calculating the minimum number of inputs per modification.

For those reasons, it is obvious that the recovery of structural controllability should not just simply identify a set of inputs. Otherwise, unnecessary computation could be paid in total, or the system might be still structurally uncontrollable in reality.

After that, the second and the last objective of this thesis is to efficiently analyze single vertices and edges of a given CT-LTI network in order to identify all vulnerable single vertices and edges to the removal. The vulnerability is in terms of maintaining structural control with a minimum set of inputs. As a result, protection on them can be implemented in advance of attacks or failures. Reasons of determining this objective are shown below:

On the one hand, given a network with CT-LTI dynamics, network vertices, which are directly forced by external inputs to structurally control the entire network, are driver nodes [40]. Once a driver node is removed, the entire network would be structurally uncontrollable immediately. Besides, attackers might hijack structural controllability by affecting identified driver nodes [33]. Even worse, nodes of all minimum set of driver nodes can have been identified by an existing method [62] in polynomial time. As a result, the first reason of doing efficient network analysis is that the diver node is easily targeted and identified by malicious attackers.

On the other hand, in terms of maintaining structural controllability with a min-

imum set of inputs, because existing related works are unable to effectively identify vulnerable single vertices and edges to the removal. The robustness of network structural controllability against node or edge removals is studied according to the vertex degree distribution, and betweenness [109, 100, 94]. In general, from numerical stimulations, continuously removing either single edges or nodes can dramatically increase the minimum number of inputs to structurally control the residual network. As a result, the residual network would be more difficult to structurally control. However, we reviewed those global graph variables to measure robustness of network structural conrollability against edge and nodal removals. We found that those variables can not indicate all vulnerable nodes and edges. As a result, protecting network controllability through those vulnerable nodes and edges by them is impossible.

Besides, there are more related works to these two objectives, which are systematically shown in section 3.2 of chapter 3.

1.3 Research Questions

In accordance of objectives and some literature reviews above, precise motivation for raising each research question is listed below, which are followed by generic assumptions and specific research questions later.

1.3.1 Dedicated Motivation

1. Because CT-LTI dynamical systems, especially for networks, could change over time due to external modification, such as removing or inserting a single vertex, whereas the harmfulness of these modification on a system vertex or an edge, might be very limited. Also, efficient recovery of controllability of each resulting network per such a modification may be essential, which requires iterative recovery of controllability. Therefore, it is helpful to recover structural controllability efficiently after a single-vertex addition, so that efficiency of entire recovery can be increased. To do this, the research question 1 is raised.

   Specifically, given equation (1.1) to represent an initially structural controllable network with a minimum set of inputs, after adding a vertex with some edges into original network, the related state matrix is obtained by adding one column and one row that contain non-zero entries into matrix $\mathbf{A}$. Then, the resulting state matrix belongs to $\mathbb{R}^{(n+1)\times(n+1)}$. With this new state matrix, that original input matrix $\mathbf{B}$ may be not proper to construct a structurally controllable system. It is thus necessary to identify another input matrix with a minimum number of columns. In particular, it is better to reuse previous inputs as many as possible.

2. Compared with control recovery after the sabotage of the communication device, controller, actuator and sensor, a resilient control systems should also efficiently restore control into physical systems after breaking physical systems. Such modification could be removal of an already targeted system component. To enhance resilience of control systems after some modifications, it

is thus useful to recover structural control into the residual physical system after removing a system component, and research question 2 is raised.

Again, given equation (1.1) to represent an originally structural controllable system with a minimum set of inputs, after removing a known vertex, the related state matrix can be obtained by removing a same indexed column and row from **A**, and it belongs to $\mathbb{R}^{(n-1)\times(n-1)}$. Then, **B** may be not proper to construct a structurally controllable system. The question is about to identify an input matrix with a minimum number of columns. Again, it is also much better to reuse previous inputs, maximally.

3. Because it can be meaningless to recover structural controllability by simply identifying a set of inputs, when recovery needs more number of inputs than that of existing ones, for example. Any possible constraints on input matrix should be concerned during the recovery of structural control into the residual system. As a result, recovery of structural controllability with given inputs by extra modification is necessary and valuable to study, and research question 3 is raised.

   In this case, given equation (1.1) to represent an originally structurally controllable system, after severe attacks or failures, accordingly, related state matrix can be obtained by removing non-zero entries and the same indexed columns and rows of **A**. Besides, constrained input matrix belongs to the same dimension as current state matrix, while it can not be changed during the entire recovery, and both of those matrices can not construct a structurally controllable system. Thus, extra modification on this resulting state matrix is the key to recover structural controllability.

4. People have used critical, redundant and ordinary categories [73] to concisely distinguish the importance of any single edge in maintaining network structural controllability with the minimum number of inputs. In detail, a removal of a critical edge gains the minimum number of inputs to structurally control residual network; removing a redundant edge never affects current minimum set of inputs; removing an ordinary link does not change the minimum number of inputs, while driver nodes that are currently used would be changed to structurally control the residual network. Nevertheless, Liu $et\ al.$ [73] used the low efficient algorithm of [97] to execute entire edge classification. Given a digraph with m arcs and n vertices in number, its worst-case execution time is $O(m^2 \cdot \sqrt{n})$.

   It is desirable to more efficiently confirm the edges of each category for further edge protection than that method above. Because a certain network analysis is still uncertain. It is therefore urgent to provide an efficient edge-classification scenario, and research question 4 is raised, in order to more efficiently distinguish the category of any given single edge than that existing result.

5. Given a digraph with m arcs and n vertices in number, because each node of all minimum set of driver nodes could be identified in $O(m \cdot n)$ steps at most. Also, since removing any single driver node directly damages current

structural controllability. It is thus necessary to efficiently identify each node of all minimum set of driver nodes of a given network with CT-LTI dynamics than [62]. In consequence, driver nodes can be under protection in advance. Besides, structural-control recovery could be caused by removing a single driver node. To more efficiently recover the minimum-input structural control against single driver node removal, it is helpful to know the harmfulness of removing a single driver node to the minimum-input structural control. Above all, research question 5 is raised.

This question is about efficiently identifying nodes of minimum sets of driver nodes in the first place. Then, it is also proposed to classify them according to the single driver-node removal on structurally controlling the residual network.

6. Because there is no sufficient focus on importance of a single node in maintaining the current minimum set of inputs to structurally control a given network, except for the node categories of [113]. Also, there are just some very limited qualitative descriptions about surge of the minimum number of inputs after continuously removing single vertices. As a result, vulnerable nodes to the removal can not be explicitly and effectively identified, let along maintain network structural controllability and enhance robustness against node removals. Therefore, research question 6 is raised.

   Research question 6 can be seen as an extension to research question 5. To solve this question, it is firstly distinguished that the importance of each single network vertex in maintaining the minimum-input structural conrollability. As a result, few nodal categories are then defined. Finally, all single network nodes are classification into those few categories.

1.3.2 Generic Assumptions

Each research question would be modeled into a graphic-theoretical problem and a network is thus defined as the input network to construct various solutions. Necessarily, some general assumptions for the input network and solution performance are specified below:

1. *The input network is a large and finite digraph, it has no self loops, parallel arcs and isolated nodes. Vertex set and arc set of the input network are not empty.*

2. *For any known set of inputs, it means that the input matrix of a state equation like equation (1.1) is known, and the state matrix is also known as well.*

3. *To assess the performance of solving any following research question, the worst-case execution time of each solution is mainly considered.*

For the other further assumptions, they would be specifically clarified during solving a precise question.

1.3.3 Questions

Combining motivations of section 1.3.1 with these assumptions above, research questions are now illustrated. In particular, each question is one-to-one corresponding to a motivation in sequence.

1. Given a network, which is structurally controllable via a known minimum set of inputs. Next, a single vertex is added into it. The question is raised as follows:

   " How to efficiently recover the minimum-input structural controllability of the resulting network without recomputation ? "

   This question is solved in chapter 4 and partially based on our publication [127].

2. Given a CT-LTI control system, which is structurally controllable via a known minimum set of inputs. Next, a known system component is removed from this system. The question is raised as follows:

   " How to efficiently recover the structural controllability of the residual system with a minimum set of inputs without recomputation ? "

   This question is solved in chapter 5 and partially based on our publication [129].

3. Given a structurally controllable CT-LTI control system, which was then experienced severe malicious attacks or failures. Next, given an input matrix that is always fixed. The question is raised as follows:

   " How to efficiently recover structural controllability of the residual system by adding a minimum set of non-zero entries into resulting state matrix ?"

   This question is solved in chapter 6 and mainly based on our publication [126].

4. Given a network, which is structurally controllable via a known minimum set of inputs. Next, to explicitly understand the importance of any single edge in maintaining structural controllability with a minimum set of inputs. The question is raised as follows:

   " How to efficiently classify its all edges into critical, redundant and ordinary categories ? "

   This question is solved in chapter 7 and partially based on our publication [130] and [128].

5. To understand the importance of any single node in maintaining a minimum set of driver nodes, a network with CT-LTI dynamics is given as an input network. The question is raised as follows:

   "Firstly, how to efficiently identify each network vertex of all minimum sets of driver nodes ?

   Secondly, investigate the importance of any single driver node in maintaining the currently used minimum set of driver nodes. "

This question is solved in chapter 8 and partially based on our publication [125].

6. Given a network, which is structurally controllable via a known minimum set of inputs. Then, to efficiently identify vulnerable single vertices to the removal. The question is raised as follows:

   " How to efficiently classify each network single vertices into few categories according to the importance of any single vertex in maintaining the current minimum number of inputs ?"

   This question is solved in chapter 9 and also partially based on our publication [128] and 7 of section 1.3.4.

These research questions are categorized into two parts, which are the part II and part III of this thesis. Part II addresses question 1, 2 and 3, focusing on efficient structural-controllability recovery. Then, the part III addresses question 4, 5, 6, paying attention to network analysis to identify vulnerable single vertices and edges to the removal in terms of maintaining structural controllability with a minimum set of inputs.

1.3.4 Publications

Above questions are solved according to our following publications:

1. Zhang, S & Wolthusen, S 2018, 'Iterative recovery of controllability via maximum matching' Paper presented at 13th IEEE International Conference on Automation Science and Engineering, Xi'an, China, 20/08/17 - 23/08/17, pp. 328-333. (DOI: 10.1109/COASE.2017.8256124)

2. Zhang, S & Wolthusen, S 2018, 'Efficient control recovery for resilient control systems' Paper presented at 15th IEEE International Conference on Networking, Sensing and Control, Zhuhai, China, 27/03/18 - 29/03/18, . (DOI: 10.1109/ICNSC.2018.8361318)

3. Zhang, S & Wolthusen, S 2018, 'Security-Aware Network Analysis for Network Controllability' Paper presented at AINA-2018 Workshops, Kracow, Poland, 16/05/18 - 18/05/18, . (DOI: 10.1109/WAINA.2018.00136)

4. Zhang, S & Wolthusen, S 2018, 'Efficient Analysis to Protect Control into Critical Infrastructures' Paper presented at 13th International Conference on Critical Information Infrastructures Security, Kaunas, Lithuania, 24/09/18 - 26/09/18, pp. 226-229. (DOI: 10.1007/978-3-030-05849-4_18)

5. Zhang, S & Wolthusen, S 2019, 'Structural Controllability Recovery via the Minimum-edge Addition' Paper presented at 2019 AMERICAN CONTROL CONFERENCE, PHILADELPHIA, United States, 10/07/19 - 12/07/19, pp. 1-6. (DOI: 10.23919/ACC.2019.8815176)

6. Zhang, S & Wolthusen, S 2019, 'Driver-Node based Security Analysis for Network Controllability' Paper presented at 17th European Control Conference

(ECC19), Naples, Italy, 25/06/19 - 29/06/19, pp. 1-6. (DOI: https://doi.org/10.23919/ECC.2019.8796264)

7. Zhang, S. Wolthusen, S. 'Efficiently Identify Vulnerable Vertices to Maintain Network Structural Controllability'. (Currently under review of IEEE Transactions on Control of Network Systems)

1.4 Contributions

Aiming at objectives of section 1.2.6, contributions of this thesis are basically summarized as follows :

1. Given an initially minimum-input structurally controllble CT-LTI system, a single component is either removed from or added into it. Then, to recover the structural control into the resulting system, an effective recovery method is designed, which can be executed in the linear time.

2. Given an initially structurally uncontrollble CT-LTI system, and a set of fixed number of inputs to recover structural controllability. Then, by adding a minimum set of edges, recovery of structural controllability is executed in more efficient polynominal time than the scenario of [37].

3. To identify all vulnerable single network edges to the removal, an initially minimum-input structurally controllable network is given as the input network. The entire network edges are more efficiently classified into critical, redundant and ordinary categories, respectively, in linear time.

   Besides, to identify all vulnerable single network vertices to the control hijack attack, an initially minimum-input structurally controllable network is given as the input network. Vertices of all minimum set of driver nodes are then more efficiently identified in linear time, and the harmfulness of removing a single driver node is also systematically explored.

   Last but not the least, to identify all vulnerable single vertices to the removal, an initially minimum-input structurally controllable network is given as the input network. Three nodal categories are the defined, and all network vertices are classified into them efficiently in linear time.

1.5 Layout

Remaining chapters of this thesis is structured as follows:

Chapter 2 is the background of this thesis. It illustrates existing knowledge that promotes us to solve all research questions of section 1.3, which includes controllability of CT-LTI, DT-LTI systems, two representative graph-theoretical methods and concepts to construct a controllable CT-LTI system. Particularly, the structural controllability [72] is primarily presented in detail. Besides, controllability of complex networks is also illustrated. This chapter qualitatively compares those graph-theoretical methods to derive completely controllable systems.

Chapter 3 reviews recent works related to solving those research questions. In one aspect, it contains the literatures about robustness of network structural controllability against both the nodal and edge removal, which are executed by various scenarios with different network types. Then, this chapter also illustrates existing works about recovering structural controllability according to the introduced methods of deriving structural controllability in section 2.4 of chapter 2. Additionally, related works that analyse network vertices and arcs for deriving structural controllability are reviewed as well. In addition to, some literatures about graph-theoretical problems are discussed, which are used to model and solve some research questions of this thesis.

Chapter 4 solves question 1. Specifically, a network is given, which is structurally controllable via a known minimum set of inputs. Next, a single vertex is added into this given network. The minimum-input structural controllability of the resulting network would be efficiently recovered by identifying a maximum matching without recomputation.

For the contribution of this chapter, given an initially structurally controllable network with a known minimum set of inputs, after a single-vertex addition on it. This chapter efficiently recovers structural controllability of the latest resulting network in linear time in the worst case. The method is about efficiently finding a maximum matching of a digraph after adding a single vertex and few arcs. Besides, previous inputs are reused with the maximum number during recovery. This result reflects that structural controllability recovery should concern the amount of change on the initial network.

Chapter 5 solves question 2. Specifically, a CT-LTI control system is given, which is structurally controllable via a known minimum set of inputs. Next, a known system component is removed from it. The structural controllability of the residual system with a minimum set of inputs is efficiently recovered by identifying a maximum matching without recomputation.

For the contribution, given a minimum-input structurally controlled CT-LTI physical system, structural-control recovery after removing a precomputed system component is executed in linear time. The method is about efficiently finding a maximum matching of a digraph after removing a known single node. Also, previous inputs are reused with the maximum number. Again, this result also reflects that structural controllability recovery should concern the amount of change on the initial system.

Chapter 6 solves question 3. Given a structurally uncontrollable CT-LTI system, this chapter efficiently recovers its structural controllability with a set of given inputs. This question is solved by adding a minimum set of edges into the network that represents this given system, so that the final digraph can represent a structurally controllable system with given inputs.

For the contribution, given a structurally uncontrollable system with fixed inputs, let m, n be the number of edges and vertices of the network representation of this given system. This chapter recovers structural control by minimum-edge addition in the worst time complexity of $O(\sqrt{n} \cdot m)$. This scenario is executed more efficient than the latest edge-addition approach of [37], whose worst-time execution time is $O(n^3)$.

Chapter 7 solves question 4. Given a structurally controllable network with

CT-LTI dynamics by a precomputed minimum set of inputs, this chapter efficiently classifies edges of this given network into critical, redundant and ordinary categories respectively by identifying all single edges that are contained by all maximum matchings based on that precomputed maximum matching.

For the contribution, given a minimum-input structurally controllable network, this chapter more sufficiently finds all maximally-matchable edges in linear time than related work of [112]. This chapter also more efficiently classifies edges of a given network than using the algorithm of [97].

Chapter 8 solves the question 5. Given a CT-LTI dynamic digraph, this chapter identifies its single vertices of all minimum sets of driver nodes more efficiently than [62]. Besides, to further understand the harmfulness caused by removing a single node of a minimum set of driver nodes, this chapter also classifies those identified nodes based on the importance of any single driver node in terms of maintaining currently used driver node set. The first issue is solved by finding each vertex that is an unmatched node related to a maximum matching of this given digraph. The next issue is solved by classifying those nodes according to impacts of single driver-node removal on the number of unmatched nodes of the residual digraph.

For the contribution of this chapter, given any CT-LTI dynamic digraph, let m, n be the number of its edges and nodes, vertices of all minimum sets of driver nodes are identified in $O(\sqrt{n} \cdot m)$. This complexity is lower than [62] in the worst case. Then, they are efficiently classified to enrich the understanding of the harmfulness caused by a single driver-node removal.

Chapter 9 solves question 6. This chapter can be seen as an expansion of the secondly solved issue of chapter 8. Given a structurally controllable network with a minimum set of inputs, this chapter effectively explores the importance of any single vertex in maintaining current minimum set of inputs. Accordingly, this chapter defines few nodal categories. Then, this chapter classifies all single vertices into those categories. For the solution, importance is confirmed by impacts of any single-node removal on the unmatched nodes of residual network related to a maximum matching. Instead of confirming the category of a single node one by one, categories of all single nodes of the given network are confirmed by running our two kinds of graph traversing and labelling algorithms once for entire network.

For the contribution, this chapter quantitatively explains the reason for surge of the minimum set of inputs during continuous removal of single vertices. Besides, this chapter also classifies all network vertices accordingly, in order to identify vulnerable vertices to the single-node removal. As a result, except for deriving the initial structural controllability with a minimum set of inputs, the worst-case execution time of entire operations is linear, which is much more efficient than iteratively recomputing a minimum set of inputs after a single node removal to confirm its category.

Chapter 10 finally summarizes this thesis, and some other things would be emphasized to guide the future work, which are still about efficient assurance of structural controllability in other different aspects.

Chapter 2

Background

2.1 Overview

This chapter illustrates existing knowledge that promotes us to solve those research questions of section 1.3 of chapter 1. It mainly elaborates controllability of continuous-time and linear time-invariant(CT-LTI) systems. Then, this chapter introduces and quantitatively compares some effective graph-theoretical methods and concepts to construct a controllable CT-LTI system. After this, structural controllability [72] is primarily presented. In terms of illustration of them, this chapter omits the process of mathematical derivation for some results. But fundamental concepts and important results would be shown and used over following thesis.

2.2 Controllability of CT-LTI systems

As mentioned in section 1.2 of chapter 1, a CT-LTI system can be described by an ordinary differential equation below, which is called the state equation [65]:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{2.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the state matrix, and its every non-zero entry illustrates which system components interact with each other and the strength of those interactions; here, any component of a system could be a physical variable, or a physical entity; $x(t) \in \mathbb{R}^n$, and $x(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T$ is the state vector, and it captures the state of each system component at time t; $u(t) \in \mathbb{R}^m$, $u(t) = (u_1(t), u_2(t), \ldots, u_m(t))^T (m \leq n)$, is the input vector, and holds external m inputs at time t; $\mathbf{B} \in \mathbb{R}^{n\times m}$ is the input matrix, and its non-zero entries show which inputs directly force which system components. Here, the state of a dynamic system is regarded as the minimal amount of information about the past history of the system, which also suffices to predict the effect of the past on the future [65].

Controllability is one of system properties, when a system is with controllability, it is completely controllable or vice versa, and controllability is defined:

Definition 2.1 (Controllability [99])
System described by equation (2.1) is controllable in a time interval $[t_0, t_f]$, if for any initial state x_0 defined at t_0 and a final state x_f defined at t_f, there are valid inputs $u_t \in \{u_t | t \in [t_0, t_f]\}$ that can steer x_0 to x_f.

Furthermore, in the research of canonical structural of linear dynamical systems [64], given the system described by equation (2.1), all system components can be classified into either controllable part or uncontrollable part, when its each system

component is controllable via inputs, or n system components are all controllable, this given dynamical system is said to be completely controllable. Also, the number of controllable components is proved to be equal to the rank of a matrix, noted by **C**, which is defined based on the state matrix **A** and the input matrix **B** by following equation (2.2):

$$\mathbf{C} = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}], (\mathbf{C} \in \mathbb{R}^{n \times nm}) \tag{2.2}$$

We use $rank(\mathbf{C})$ to represent the rank of **C**. Also, matrices $\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}$ are n submatrices of matrix **C** in number, and each of which belongs to $\mathbb{R}^{n \times m}$. Therefore, when a system described by equation (2.1) is completely controllable, if and only if matrix **C** has full rank, which is called the controllability rank condition, and noted by:

$$rank(\mathbf{C}) = n \tag{2.3}$$

With controllability rank condition, to construct a completely controllable CT-LTI system according to a given state matrix, it actually needs to identify an input matrix, so that the controllability rank condition is satisfied. Also, to verify if a given CT-LTI system is completely controllable or not, it just needs to calculate if this system satisfies the controllability rank condition by equation (2.2) and (2.3).

2.2.1 Controllability of DT-LTI systems

As mentioned in section 1.2.2 of chapter 1, the algebraic condition of a CT-LTI system and that of a discrete-time and linear time-invariant (DT-LTI) system are completely same, now this section formally proves this statement.

Generally, a DT-LTI system is described by the following equation:

$$x(t) = \mathbf{A}x(t-1) + \mathbf{B}u(t-1)(t \geq 1) \tag{2.4}$$

where state matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, input matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$, state vector $x(t) \in \mathbb{R}^n$ and input vector $u(t) \in \mathbb{R}^m$ here, are completely same as that of equation (2.1). Firstly, a completely controllable DT-LTI system can be intuitively defined:

Definition 2.2 (Controllability of a DT-LTI System [19])
A system described by equation (2.4) is completely controllable in (n - 1)-time steps, when there exists a sequence of inputs $u(1), u(2) \ldots u(n-1)$, such that $x(n)$ is the obtained, and it is the proposed state vector regardless of the initial state $x(1)$.

Given a system described by equation (2.4), and a sequence of inputs $\{u(1), u(2) \ldots u(n-1)\}$ for $(n-1) - time$ steps. Then, with initial state $x(1)$, if the given system is completely controllable in $(n-1) - time$ steps, there are following $(n-1)$ equations:

$$\begin{aligned} x(2) &= \mathbf{A}x(1) + \mathbf{B}u(1) \\ x(3) &= \mathbf{A}x(2) + \mathbf{B}u(2) \\ x(4) &= \mathbf{A}x(3) + \mathbf{B}u(3) \\ &\vdots \\ x(n) &= \mathbf{A}x(n-1) + \mathbf{B}u(n-1) \end{aligned}$$

That final equation can be further written by a following one, which represents the relationship between state $x(n)$ and state $x(1)$ in detail:

$$x(n) - \mathbf{A}^{(n-1)} \cdot x(1) = \sum_{r=1}^{n-1} \Big(\mathbf{A}^{r-1} \cdot \mathbf{B} \cdot u(n-r) \Big) \tag{2.5}$$

Then, let $x(n) = \mathbf{0}$, and this equation can be also written as:

$$-\mathbf{A}^{(n-1)} \cdot x(1) = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{(n-1)-1}\mathbf{B}] \cdot \begin{bmatrix} u(n-1) \\ u(n-2) \\ u(n-3) \\ \vdots \\ u(1) \end{bmatrix} \tag{2.6}$$

because this given DT-LTI system is controllable, there must be a solution for this equation, and matrix $[\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{(n-1)-1}\mathbf{B}] \in \mathbb{R}^{n \times (m \times (n-1))}$ must be invertible and it is thus full rank. By contrast, this result is equivalent to the rank condition of a CT-LTI system that is represented by equation (2.2). Thus, when a DT-LTI system is completely controllable, controllability rank condition for CT-LTI system is also applicable.

Reversely, if matrix $[\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{(n-1)-1}\mathbf{B}]$ is full rank, given $(n-1)$ inputs $\{u(1), u(2) \ldots, u(n-1)\}$ and a system described by equation (2.4). Considering equation (2.6), because of the same dimension of those two matrices at right and left side of equal mark, which is $\mathbb{R}^{n \times (m \times (n-1))}$. Those $(n-1)$ equations can be also deduced, which indicate that this system is controllable in $(n-1) - time$ steps with these inputs in consequence.

Therefore, controllability rank condition for CT-LTI system is always applicable for design and analyse DT-LTI systems.

2.3 Structural Controllability

Although controllability rank condition shown by equation (2.2) and (2.3) of section 2.2, offers a comprehensive, rigorous and detailed framework for the design and analysis of CT-LTI systems, there are two general problems preventing against its usage in practice. Given a system described by equation (2.1), one problem is that non-zero entries of both matrix **A** and **B** are known with approximation of some errors of measurement [72]. Another problem is that the computation of $rank(\mathbf{C})$ is massive, which requires $2^n - 1$ different combinations in the worst case [73].

To avoid these two problems and still design or analyze controllable CT-LTI systems, the structured system is concerned. Generally speaking, for any given system described by equation (2.1), it is a structured system, if entries of both **A** and **B** are either fixed zeros or independent free parameters [49], where such **A** and **B** are called the structured matrices. Besides, given a structured system, one of its instances is the system, whose non-zero entries of both state and input matrices are exact values. Then, researchers use a structured system to explore when an instance of it is completely controllable, or how to derive a completely controllable instance of a structured system. In recent four decades, for this purpose, various graph-theoretical methods have been raised, and some representative methods and

concepts are introduced in section 2.3-2.5, of which, the structural controllability and maximum-matching based method are mainly concentrated in following other chapters.

Firstly, given a state matrix, it is valid to construct a CT-LTI system described by equation (2.1) via the structural controllability, so that it is very likely that a completely controllable system can be obtained. Lin *et al.* [72][104] initially raised structural controllability, and it is defined below:

Definition 2.3 (Structural Controllability [72])
Given a structured system described by equation (2.1), then, this structured system is structurally controllable, if and only if there is at least one instance of it satisfying the controllability rank condition.

According to this definition, given any structurally controllable system, there must be at least one completely controllable instance of it, and it can be observed that structurally controllable system is a necessary but not sufficient condition of a completely controllable system. Even so, the structural controllability is a powerful concept, because it is proved that once an instance of a structured system satisfies the controllability of rank condition, almost all instances of a same structured system are completely controllable [45]. In other words, given a structurally controllable system described by equation (2.1), any instance of it should be completely controllable with high probability. Further more, conditions of a structurally controllable system are generalized by theorem 2.1 with following definitions:

Definition 2.4 (Digraph G(A, B))
Given matrix $\boldsymbol{A}$, $\boldsymbol{B}$ of equation (2.1), let $G(\boldsymbol{A},\boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ be a digraph, where $V_1 = \{v_i|1 \leq i \leq n\}$ and $V_2 = \{u_j|1 \leq j \leq m\}$ are two vertex sets, and $E_1 = \{\overrightarrow{\langle v_i, v_k\rangle}|v_i, v_k \in V_1\}$ and $E_2 = \{\overrightarrow{\langle u_j, v_i\rangle}|u_j \in V_2, v_j \in V_1\}$ are two edges sets. Also, let $\alpha : \{\boldsymbol{A},\boldsymbol{B}\} \to G(\boldsymbol{A},\boldsymbol{B})$ be a bijection. For each non-zero $a_{ki} \in \boldsymbol{A}$ and $b_{ij} \in \boldsymbol{B}$, there are $\alpha : a_{ki} \to \overrightarrow{\langle v_i, v_k\rangle}$ and $\alpha : b_{ij} \to \overrightarrow{\langle u_j, v_i\rangle}$.

Remark 1 In this thesis, sometimes, digraph mapped by a given system state equation according to definition 2.4, is called the system network of a pair of given state matrix and input matrix. □

Definition 2.5 (Stem & Bud[72])
Given $G(\boldsymbol{A},\boldsymbol{B})$ of definition 2.4, a stem of it is a directed path starting from a node of V_2. A bud is a directed cycle plus an arc, whose head is shared with this cycle, and this arc is called the distinguished edge.

Definition 2.6 (Dilation of a Digraph[72])
Given $G(\boldsymbol{A},\boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ of definition 2.4, $S \subseteq V_1$ is a set of nodes, $T(S) \subseteq V_1 \cup V_2$ is a set of vertices as tails of arcs whose heads are all vertices of S. Then, $G(\boldsymbol{A},\boldsymbol{B})$ contains a dilation, if and only if $|S| > |T(S)|$.

Definition 2.7 (Inaccessibility [72])
Given $G(\boldsymbol{A},\boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ of definition 2.4, and a node of V_1. Then, this single node is inaccessible, if and only if it can not be visited through directed paths that start from any node of V_2 .

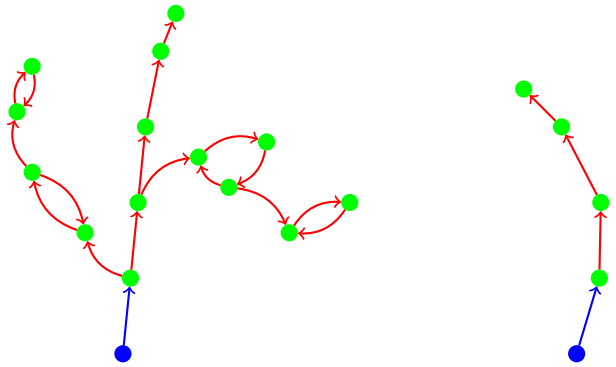

Figure 2.1: Cacti

According to definition 2.8, here is a set of disjoint cacti, where only the blue vertices are mapped by inputs.

Definition 2.8 (Cactus[72])
Let $B_1, B_2, \ldots, B_l$ be a set of buds, and let S_1 be a stem, $S_1 \cup B_1 \cup B_2, \ldots, \cup B_l$ is a cactus if and only if the tail of distinguished edge of $B_i (1 \leq i \leq l)$ is not the top node of S_1 but the only common node of $S_1 \cup B_1 \cup B_2, \ldots \cup B_{i-1}$. Besides, a stem of definition 2.5 is also a cactus. (see an example in figure 2.1.)

Theorem 2.1 (Structural Controllability Theorem [72, 79])
Given system described by equation (2.1), and $G(\boldsymbol{A}, \boldsymbol{B})$ of definition 2.4, then, following statements are equivalent:

1. *System described by equation (2.1) is structurally controllable.*
2. *$G(\boldsymbol{A}, \boldsymbol{B})$ contains neither inaccessible nodes nor a dilation.*
3. *$G(\boldsymbol{A}, \boldsymbol{B})$ is spanned by a set of disjoint cacti.*

In particular, according to [72, 79], when statement one is satisfied, statement two implied by it can thus imply statement three, while statement three can always imply both statement one and two. It means that the second statement can not be independently used to verify if a system is structurally controllable or not. For example, given a CT-LTI system described by the following state equation (2.7), and it is mapped into the digraph of figure 2.2 according to definition 2.4. As we can observe, this digraph has neither the dilation nor the inaccessible vertices from u_1. But, this digraph is not spanned by a cactus.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & 0 & a_{45} \\ a_{51} & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot u(t) \tag{2.7}$$

Thus, we conclude corollary 2.2 to clarify such relationship among this three statements of theorem 2.1.

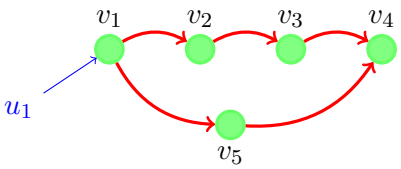

Figure 2.2: A digraph of the system of equation (2.1) by definition 2.4.

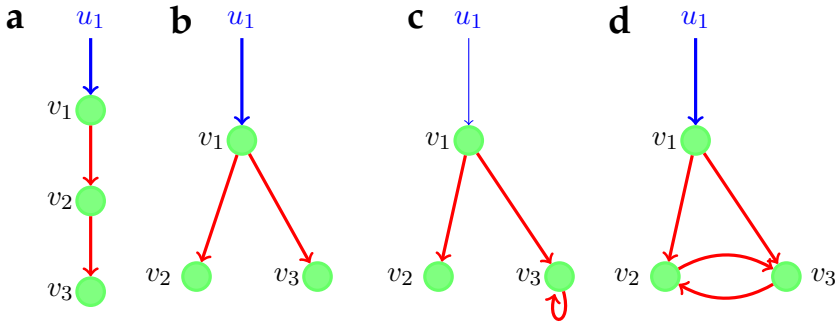

Figure 2.3: Digraphs mapped by four simple CT-LTI systems.

Corollary 2.2
By theorem 2.1, a system described by equation (2.1) is structurally controllable if and only if the network mapped from it by definition 2.4 not only excludes inaccessible nodes and a dilation, but also spanned by a set of disjoint cacti.

2.3.1 Simple Examples

This section shows some simple examples to visually clarify the relationship between structural controllability and controllability of CT-LTI systems. Four digraphs of figure 2.3 are mapped by four CT-LTI systems according to the bijection of definition 2.4, and the controllability and structural controllability of each related system are discussed.

1. For figure 2.3 **a**, the related system can be written as:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ 0 & a_{32} & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \\ 0 \end{bmatrix} \cdot u(t)$$

and the controllability matrix is thus obtained by:

$$\mathbf{C} = [\mathbf{B}, \mathbf{A} \cdot \mathbf{B}, \mathbf{A}^2 \cdot \mathbf{B}] = \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_1 \cdot a_{21} & 0 \\ 0 & 0 & b_1 \cdot a_{32} \cdot a_{21} \end{bmatrix}$$

By theorem 2.1, this system is structurally controllable due to there is only a stem of definition 2.5. Also, since $rank(\mathbf{C}) = 3$ is constant, all instances of this system are always completely controllable.

2. For figure 2.3 **b**, the related system can be written as:

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ a_{21} & 0 & 0\\ a_{31} & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\end{bmatrix} + \begin{bmatrix}b_1\\ 0\\ 0\end{bmatrix} \cdot u(t).$$

Because the digraph is not spanned by a cacti of definition 2.8, this system is not structurally controllable. Also, since structurally controllable system is the necessary but not sufficient condition of a completely controllable system, all instances of this system represented by figure (**b**) are of course not completely controllable.

3. For figure 2.3 **c**, the related system can be written as:

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ a_{21} & 0 & 0\\ a_{31} & 0 & a_{33}\end{bmatrix} \cdot \begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\end{bmatrix} + \begin{bmatrix}b_1\\ 0\\ 0\end{bmatrix} \cdot u(t)$$

and the controllability matrix is thus obtained by:

$$\mathbf{C} = [\mathbf{B}, \mathbf{A}\cdot\mathbf{B}, \mathbf{A}^2\cdot\mathbf{B}] = \begin{bmatrix}b_1 & 0 & 0\\ 0 & b_1\cdot a_{21} & 0\\ 0 & b_1\cdot a_{31} & b_1\cdot a_{33}\cdot a_{31}\end{bmatrix}$$

Obviously, since $rank(\mathbf{C}) = 3$, all instances of this system are thus completely controllable and thus structurally controllable.

4. For figure 2.3 **d**, the related system can be written as:

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ a_{21} & 0 & a_{23}\\ a_{31} & a_{32} & 0\end{bmatrix} \cdot \begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\end{bmatrix} + \begin{bmatrix}b_1\\ 0\\ 0\end{bmatrix} \cdot u(t)$$

and the controllability matrix is thus obtained by:

$$\mathbf{C} = [\mathbf{B}, \mathbf{A}\cdot\mathbf{B}, \mathbf{A}^2\cdot\mathbf{B}] = \begin{bmatrix}b_1 & 0 & 0\\ 0 & b_1\cdot a_{21} & b_1\cdot a_{23}\cdot a_{31}\\ 0 & b_1\cdot a_{31} & b_1\cdot a_{32}\cdot a_{21}\end{bmatrix}$$

Although the digraph is spanned by a stem starting from u_1, once the value of each non-zero entry of structured matrices is equal to 1, then $rank(\mathbf{C}) = 2 < 3$. Clearly, this system is structurally controllable, but not all instances of it are completely controllable.

Combining these examples with theorem 2.1, controllability of a given CT-LTI system can be obtained or analysed by structural controllability with the graphic interpretation of it. Next, two related methods to derive structural controllability are systematically shown in section 2.4.

2.4 Derive Structural Controllability

In general, based on theorem 2.1, given a state matrix **A** of equation (2.1), constructing a CT-LTI system containing it with the structural controllability, can be modelled into a graph problem. Specifically, derive a digraph $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4 in the beginning, which is mapped by the given state matrix. Next, identify a vertex set V_2 and an edge set E_2 of definition 2.4, whose vertices are adjacent to nodes of V_1 and as tails of arcs of E_2. Finally, the resulting digraph $(V_1 \cup V_2, E_1 \cup E_2)$ should be spanned by a set of disjoint cacti of definition 2.8. As a result, the input matrix **B** can be obtained by mapping V_2 and E_2 according to the bijection of definition 2.4. After that, a structurally controllable CT-LTI system is derived, eventually. Accordingly, given $G(\mathbf{A}) = (V_1, E_1)$, two different graph-theoretical methods to identify V_2 and E_2 of definition 2.4 are shown in section 2.4.1 and 2.4.2, respectively.

2.4.1 Maximum-Matching based Method

Definition 2.9 (Maximum Matching of a Digraph[35, 14])
In a digraph, a matching is a set of arcs without common tails and heads, and a maximum matching is a matching with the maximum cardinality. For any single vertex, it is an unmatched node with respect to a matching, if and only if it is not the head of any arc of this matching. Otherwise, it is matched.

Definition 2.10 (Maximum Matching of a Bipartite or Undirected Graph [35, 14])
In a bipartite graph or an undirected graph, a matching is a set of vertex-disjoint edges, and a maximum matching is a matching with the maximum cardinality. For a single vertex, it is an unmatched node with respect to a matching, if and only if it is not incident to any edge of this matching. Otherwise, it is matched.

Definition 2.11 (Perfect Matching[35, 14])
In a graph, which can be directed, undirected, or bipartite, when all vertices are matched with respect to a maximum matching, this graph is said to have a perfect matching.

Generally, there are multiple maximum matchings within a same graph. Given a graph and a maximum matching of it, this maximum matching with those unmatched nodes, can span this given graph. To identify a maximum matching, the augmenting path of definition 2.12, plays a critical role by following corollaries:

Definition 2.12 (Augmenting Path [60])
Let B_0 be a bipartite graph, E_0 be the edge set of B_0, and M_0 be a matching of B_0. With respect to M_0, an alternating path is a path alternatively involving edges of M_0 and $E_0 \setminus M_0$. An alternating path is an augmenting path with respect to M_0, if and only if its two terminals are not incident to any edge of M_0. (See examples in figure 2.4.)

Corollary 2.3
Let B_0 be a bipartite graph, M_0 be an arbitrary given matching of B_0, and P_{aug} be an existing augmenting path with respect to M_0. Then, the symmetric difference between P_{aug} and M_0, noted by $P_{aug} \oplus M_0$ and $P_{aug} \oplus M_0 = \{P_{aug} \setminus \{P_{aug} \cap M_0\}\} \cup \{M_0 \setminus \{P_{aug} \cap M_0\}\}$, is a matching bigger than M_0 by one in cardinality [89, 20].

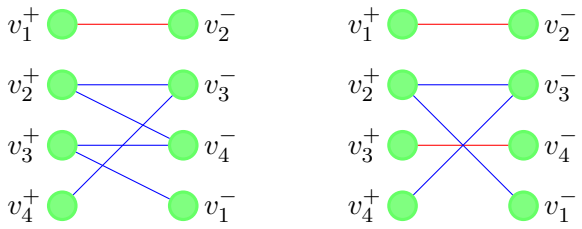

Figure 2.4: Examples of Augmenting Paths

In two bipartite graphs, with respect to a matching $\{(v_2^-, v_1^+), (v_3^-, v_2^+), (v_4^-, v_3^+)\}$, each complete blue path is an augmenting path.

Corollary 2.4
Let B_0 be a bipartite graph, and M_0 be an arbitrary matching of B_0, M_0 is a maximum matching if and only if no augmenting path related to M_0 exists [89, 20].

Therefore, based on these two statements, a maximum matching of a bipartite graph can be identified by iteratively identifying an augmenting path related to each latest-identified matching. In detail, given a bipartite graph with m edges and n vertices in the number, identifying an augmenting path related to the latest-identified matching costs $O(m)$ steps at most, while the number of iterations is $O(\sqrt{n})$ maximally [60]. Thus, worst-case execution time of identifying a maximum matching of a bipartite graph is $O(\sqrt{n} \cdot m)$.

Additionally, because any directed graph can be mapped into a bipartite graph in linear time, where one vertex set of this bipartite graph corresponds to a set of all vertices as tails of arcs of the digraph, and another vertex set corresponds a set of all nodes as heads of arcs of the digraph [86]. Therefore, a maximum matching of this digraph can be identified through a maximum matching of a bipartite graph mapped by it. An example of the mapping between a digraph and a bipartite graph is shown in figure 4.1 with bijection of definition 4.2 of chapter 4.

Besides, given a general graph with m edges and n vertices in the number, a maximum matching can be also identified in $O(m \cdot \sqrt{n})$ steps at most [81]. Up to date, this result is the bound of efficiently finding maximum matching by deterministic algorithms except for approximation, which could be executed much faster than identifying an exact maximum matching on static graphs [57]. When the given bipartite graph is dense such as $m \geq n^2$, time complexity becomes $O(m\sqrt{n/logn})$ [8]. Besides, for identifying a maximum matching of a sparse *Erdős-Rényi* random graph [51], the average case time complexity is just $O(m \cdot log(n))$ [16].

2.4.1.1 An Algorithm

Above, we show how to use a maximum matching to obtain a cacti, so that structural controllability can be obtained. Here, given $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, let M_D be a maximum matching of it, V_{1M_D} be a set of nodes of V_1 that are unmatched related to M_D, and c_j be a cycle of M_D. Also, let v_p, v_q be any two different vertices of V_1. Let $V_2 \neq V_1$ and $E_2 \neq E_1$ be the initial empty vertex and arc set, and $u_i \notin V_1$ be a vertex out of V_1. Then, algorithm 2.1 below shows how to use M_D to identify V_2 and E_2 so that $(V_1 \cup V_2, E_1 \cup E_2)$ is spanned by a set of disjoint cacti.

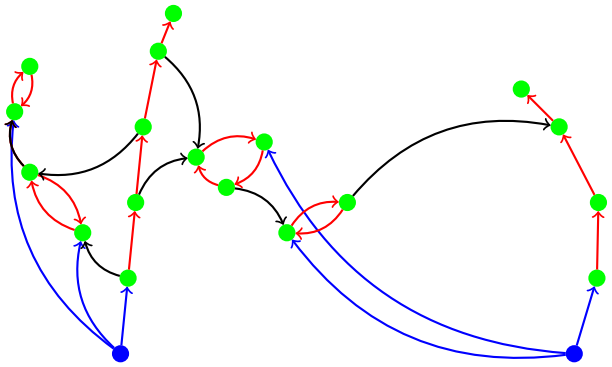

Figure 2.5: A Maximum Matching & A Cacti

The digraph only involves all green nodes, black and red arcs, which is mapped by a given state matrix, where all red arcs construct a maximum matching. Then, two blue nodes represents two identified inputs, and the blue arcs are either incident to unmatched nodes or a node of a cycle. Finally, all green and blue nodes, red arcs and blue arcs construct two disjoint cacti by algorithm 2.1.

Algorithm 2.1: Identify V_2 and E_2 by a maximum matching of $G(\mathbf{A})$.

Input : $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, V_2, E_2.
Output: V_2, E_2.

Identify M_D of $G(\mathbf{A}) = (V_1, E_1)$ by the algorithm of [81] or [60];
Identify V_{1M_D}; Identify each $c_j \subseteq M_D$ by algorithm of [111];
while $V_{1M_D} \neq \emptyset$ ***and*** $v_p \in V_{1M_D}$ **do**
    $V_{1M_D}^{'} = V_{1M_D} \setminus v_p$;
    $V_2^{'} = V_2 \cup u_i$;
    $E_2^{'} = E_2 \cup \overrightarrow{\langle u_i, v_p \rangle}$;
while *for each* c_j ***and*** $v_q \in c_j$ **do**
    $M_D^{'} = M_D \setminus c_j$;
    For any $u_i \in V_2$, $E_2^{'} = E_2 \cup \overrightarrow{\langle u_i, v_q \rangle}$;
return V_2, E_2;

PROOF Because M_D contains disjoint directed paths and cycles, $V_{1M_D} \cup M_D$ thus spans $G(\mathbf{A})$. Also, when each node of V_{1M_D} is adjacent to a node of V_2 ($|V_2| = |V_{1M_D}|$) and as the head of E_2, stems are produced, which is done by the first **while** loop. Then, when any vertex of each cycle of M_D is adjacent to a node of V_2 and as a head of E_2, a set of disjoint cacti is generated, finally, which is done by the second **while** loop. Thus, algorithm 2.1 is correct. Besides, identifying each cycle of M_D, V_{1M_D} and running these two **while** loops totally cost $O(|E_1| + |V_1|)$ steps at most. Time complexity of running this algorithm in the worst case is $O(|E_1| \cdot \sqrt{|V_1|})$, so that the input matrix for a given state matrix can be obtained, and a structurally controllable system containing the given state matrix can be effectively derived in polynomial time. An example about this algorithm is shown in figure 2.5.

In this example, although there is a set of disjoint cacti of figure 2.1, another cacti is also obtained by a maximum matching. ■

Besides, V_2 identified by algorithm 2.1 is with the minimum cardinality, because

cardinality of V_2 is determined by a minimum set of unmatched nodes related to a maximum matching, and a statement about constructing a structurally controllable system with minimum set of inputs can be concluded:

Corollary 2.5
Given a state matrix $\boldsymbol{A}$ of equation (2.1), then by a maximum matching of digraph $G(\boldsymbol{A}) = (V_1, E_1)$ of definition 2.4, to construct a structurally controllable CT-LTI system involved $\boldsymbol{A}$, the minimum number of inputs is equal to one, if this maximum matching is a perfect matching. Otherwise, it equals to the number of unmatched nodes related to this maximum matching.

PROOF The correctness of this corollary can be proved by the correctness of algorithm 2.1. ∎

Simultaneously, the maximum-matching based method can be a useful tool to analyse controllability of CT-LTI systems, because any structurally controllable CT-LTI system is very likely to be completely controllable [45]. Thus, in the view of assuring or securing structural controllability of CT-LTI systems, it is essential to use more efficient method than simply recomputing a maximum matching to obtain and analyse structural controllability, so that the structural controllability can be efficiently assured or protected before or after malicious attacks or random failures. For those works about structural-controllability assurance, they are completed from chapter 4 to chapter 9.

2.4.2 Method based on Power Dominating Set

In addition to the maximum-matching based method introduced in section 2.4.1, given the digraph $G(\mathbf{A}) = (V_1, E_1$ of definition 2.4 again, in order to also derive V_2 and E_2 of definition 2.4, an alternative method that is based on the power dominating set of $G(\mathbf{A})$ would be shown in this section. As a result, digraph $(V_1 \cup V_2, E_1 \cup E_2)$ spanned by a set of disjoint cacti can be also obtained, and a structurally controllable system is rendered. In the beginning, some fundamental terminologies are defined below:

Definition 2.13 (A Dominating Set of a Graph [55])
Let $G = (V, E)$ be an undirected graph, and $V_{sub} \subseteq V$ be a subset of V. Then, V_{sub} is a dominating set of G if each vertex of V is either in V_{sub}, or adjacent to a vertex of V_{sub}. Besides, a vertex is said to dominate itself and all its neighbours.(see an example in figure 2.6).

Definition 2.14 (Domination Number [55])
Let $G = (V, E)$ be an undirected graph, the domination number of G is the minimum cardinality of a dominating set of G, which is noted by $\gamma(G)$. And a minimum dominating set of G is noted by $\gamma(G) - set$.(see an example in figure 2.6).

The basic decision problem of a minimum dominating set is NP-complete with a polynomial-time approximation factor $\Theta(log(n))$ [52], where n is the number of vertices of the given graph. According to the dominating set, let $G = (V, E)$ be an undirected graph, v be a vertex of V, and $\delta(v)$ be a set of neighbours of v. Haynes *et al.* [59] raised the power dominating set according to two following observation rules:

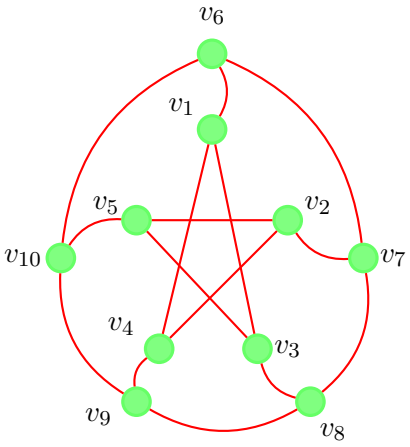

Figure 2.6: A Minimum Dominating Set of an Undirected Graph

In this graph, one of minimum dominating sets is $\{v_7, v_1, v_{10}\}$, and its domination number is thus 3.

Definition 2.15 (Observation Rules of undirected graphs [67])

1. *a vertex of a power dominating set observes itself and all of its neighbours.*
2. *if an observed vertex $v \in V$ with $|\delta(v)| \geq 2$ is adjacent to $|\delta(v)|-1$ observed vertices by nodes different from v, the unobserved neighbour of v becomes observed by v as well.*

Definition 2.16 (A Power Dominating Set of an Undirected Graph [59])
Let $G = (V, E)$ be an undirected graph, $P \subseteq V$ be a vertex set. Then, based on two observation rules of definition 2.15, P is a power dominating set of G if vertices of P observe all nodes of V. Also, by definition 2.14 the power domination number of G is the minimum cardinality of P, and it is also noted by $\gamma(G)$.

Above all, the observation rules are modified for a digraph, and a power dominating set of a digraph is defined:

Definition 2.17 (Observation Rules of Digraphs)
Let $D = (V_D, E_D)$ be a digraph, v_D be a vertex of V_D, and $\delta_{out}(v_D)$ be a set of nodes that are heads of arcs of E_D and whose tails are v_D.

1. *any v_D of a power dominating set of D observes itself and all nodes of $\delta_{out}(v_D)$.*
2. *if v_D is observed, $|\delta_{out}(v_D)| \geq 2$, and $|\delta(v)| - 1$ vertices of $\delta_{out}(v_D)$ are observed by nodes different from v_D. The unobserved node of $\delta_{out}(v_D)$ becomes observed by v_D as well.*

Definition 2.18 (A Power Dominating Set of a Digraph)
Let $D = (V_D, E_D)$ be a digraph, $P_D \subseteq V_D$ be a vertex set. Then, P_D is a power dominating set of D, if vertices of P_D observe all nodes of V_D. And the power domination number of D is the minimum cardinality of P_D, and it is also noted by $\gamma(D)$.(see example of figure 2.7).

Viewing figure 2.7, a power dominating set can actually observe all vertices in a given digraph through directed paths of the given digraph. These paths all

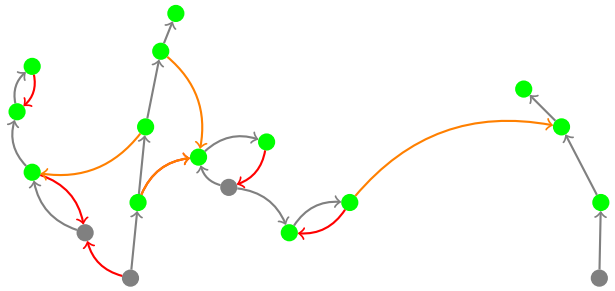

Figure 2.7: A Power Dominating Set of a Digraph

This digraph is obtained by partially citing the cacti of figure 2.5. Here all gray vertices are a set of power dominating set, and all gray arcs are used to observe all nodes of this digraph by observation rules of definition 2.17.

start from vertices of an identified power dominating set, and might only share the common staring vertices, which depends on the number of out degrees of the vertex of this chosen power dominating set. Furthermore, a set of disjoint directed paths can be thus obtained, which can be used to confirm a set of disjoint stems of definition 2.5 with extra vertices out of the given digraph, so that the final digraph can be spanned by a set of disjoint cacti. Simultaneously, this final digraph thus represent a structurally controllable CT-LTI system according to theorem 2.1.

2.4.2.1 Algorithms

Next, given digraph $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, algorithm 2.2 identifies a power dominating set of $G(\mathbf{A})$ according to definition 2.17 and 2.18. Here, let $P_{G(\mathbf{A})}$, $P_{0_{G(\mathbf{A})}}$ be two initially empty vertex sets, and $E_{P_{G(\mathbf{A})}}$ be an initially empty edge set. Also, let e_i, e_j be any two arcs of E_1, and let v_i, v_j, v_k be any three vertices of V_1, where $v_j \in P_{0G(\mathbf{A})}$, and v_i is the tail of e_i, $if P_{0G(\mathbf{A})} \neq \emptyset$.

Algorithm 2.2: Identify a power dominating set of $G(\mathbf{A})$.

Input : $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, $P_{G(\mathbf{A})}, E_{P_{G(\mathbf{A})}}$.
Output: $P_{G(\mathbf{A})}, E_{P_{G(\mathbf{A})}}$.

while $\exists v_i \in V_1$ *not coloured* **do**
  Colour v_i into red **and** $P'_{G(\mathbf{A})} = P_{G(\mathbf{A})} \cup v_i$;
  while $e_i \in E_1$ ***and*** *head of* e_i *not coloured* **do**
    $E'_{P_{G(\mathbf{A})}} = E_{P_{G(\mathbf{A})}} \cup e_i$;
    Colour head of e_i into red **and** Add head of e_i into $P_{0G(\mathbf{A})}$;
  while $P_{0G(A)} \neq \emptyset$ ***and*** $v_j \in P_{0G(A)}$ **do**
    $P'_{0G(\mathbf{A})} = P_{0G(\mathbf{A})} \setminus v_j$;
    if $\exists \overrightarrow{\langle v_j, v_k \rangle} \in E_1$ ***and*** v_k *not coloured* **then**
      $E'_{P_{G(\mathbf{A})}} = E_{P_{G(\mathbf{A})}} \cup \overrightarrow{\langle v_j, v_k \rangle}$;
      Colour v_k into red;
      $P'_{0G(\mathbf{A})} = P_{0G(\mathbf{A})} \cup v_k$;
return $P_{G(\mathbf{A})}, E_{P_{G(\mathbf{A})}}$;

PROOF In this algorithm, $P_{G_{\mathbf{A}}}$ is a power dominating set of $G(\mathbf{A})$, and $E_{p_{G(\mathbf{A})}}$ is a set of arcs used to observe all vertices of $G(\mathbf{A})$. In detail, the first **while** loop of line 1 initially chose a vertex as the first node of a power dominating set. Then, the second **while** loop of line 3 adds all arcs whose tails are v_i into $E_{P_{G(\mathbf{A})}}$ according to the first rule of definition 2.17. Also, heads of all added arcs are added into $P_{0G(\mathbf{A})}$ to keep observing nodes of $G(\mathbf{A})$ according to the second rule of definition 2.17, which is done by the last **while** loop of line 6. In this loop, because each node of $P_{0G(\mathbf{A})}$ should only observe one vertex that is not coloured, so that the second rule of definition 2.17 can be satisfied. For this purpose, once $\overrightarrow{\langle v_j, v_k \rangle}$ is added into $E_{P_{G(\mathbf{A})}}$, v_j is removed from $P_{0G(\mathbf{A})}$. After running the third **while** loop, we can obtain a set of directed paths starting from v_i chosen in line 1. Therefore, the first iteration of **while** loop of line 1 is correct. Then, since any node observed is coloured, there would not be a vertex observed twice. And also, there would be a moment that all nodes of $G(\mathbf{A})$ are coloured and this algorithm terminates. This algorithm is correct to identify a power dominating set. Additionally, due to each arc and vertex of $G(\mathbf{A})$ are coloured and visited once only, this algorithm runs in $O(|V_1| + |E_1|)$ steps at most. In particularly, this power dominating set is not necessarily minimum. ■

With $P_{G(\mathbf{A})}$ and $E_{P_{G(\mathbf{A})}}$ returned by algorithm 2.2, given $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, let $V_2 \neq V_1$ and $E_2 \neq E_1$ be the initial empty vertex and arc set, and u_i, u_j be two arbitrary vertices out of V_1. The following algorithm 2.3 uses $P_{G(\mathbf{A})}$ and $E_{P_{G(\mathbf{A})}}$ to identify V_2 and E_2 so that $(V_1 \cup V_2, E_1 \cup E_2)$ is finally spanned by a set of disjoint stems. Besides, let v_i be any vertex of $P_{G(\mathbf{A})}$, and $\delta_{out}(v_i)$ be a set of nodes as heads of arcs whose tails are v_i of digraph $(P_{G(\mathbf{A})}, E_{P_{G(\mathbf{A})}})$, and let v_j be any node of $\delta_{out}(v_i)$.

Algorithm 2.3: Identify V_2 and E_2 by a power dominating set of $G(\mathbf{A})$.

Input : $P_{G(\mathbf{A})}$, $E_{P_{G(\mathbf{A})}}$ returned by algorithm 2.2, V_2, E_2.
Output: V_2, E_2.

1 **while** *for each* v_i *of* $P_{G(A)}$ **do**
2 $\quad V_2^{'} = V_2 \cup u_i$;
3 $\quad E_2^{'} = E_2 \cup \overrightarrow{\langle u_i, v_i \rangle}$;
4 $\quad P_{G(\mathbf{A})}^{'} = P_{G(\mathbf{A})} \setminus v_i$;
5 $\quad$ **while** $|\delta_{out}(v_i)| > 1$ ***and*** $v_j \in \delta_{out}(v_i)$ **do**
6 $\quad\quad V_2^{'} = V_2 \cup u_j$;
7 $\quad\quad E_2^{'} = E_2 \cup \overrightarrow{\langle u_j, v_j \rangle}$;
8 $\quad\quad \delta_{out}(v_i)^{'} = \delta_{out}(v_i) \setminus v_j$;
9 **return** V_2, E_2;

PROOF By algorithm 2.2, since $P_{G(\mathbf{A})}$ and $E_{P_{G(\mathbf{A})}}$ is a set of directed paths that span $G(\mathbf{A})$ and share nodes of $P_{G(\mathbf{A})}$ as common starting vertices. Thus, algorithm 2.3 sets an element for V_2 and E_2 according to $P_{G(\mathbf{A})}$. For any $v_i \in P_{G(\mathbf{A})}$ chosen in line 1, since it has been a root of a stem, its neighbours of $\delta_{out}(v_i)$ with cardinality of $|\delta_{out}(v_i)| - 1$ are used to one-to-one set other elements of V_2 and E_2, so that a set

of disjoint stems starting from nodes of V_2 are obtained before the first iteration. Then, following nodes of remaining $P_{G(\mathbf{A})}$ are chosen to set V_2 and E_2 as before. When $P_{G(\mathbf{A})} = \emptyset$ due to line 4, this algorithm terminates. Therefore, this algorithm is correct. Also, since any single vertex of $P_{G(\mathbf{A})}$ and $E_{P_{G(\mathbf{A})}}$ is chosen once only, this algorithm runs in linear time. ■

To obtain less number of inputs by these algorithms, it is better to identify a power dominating set with the minimum cardinality. In the research of Haynes *et al.* [59], it has been proved that identifying a power dominating set with the minimum cardinality of a given undirected graph is a NP-complete problem, where this conclusion is still valid for some certain classes of graphs, such as bipartite graphs, chordal graphs. For cubic graphs, identify such a power dominating set is a NP-hard problem [24]. Similarly, it is also a NP-complete problem to identify a power dominating set with the minimum cardinality of a given digraph. Also, Aazami *et al.* [1] proved that a minimum power dominating set of a given digraph can not be approximated less than a threshold. And, they showed that this problem can be optimally solved in linear time if the underlying undirected graph of the given digraph has bounded tree-width.

Concerning the worst-case execution time of both graph-theoretical method, the maximum-matching based method of section 2.4.1 is applied to solve some research questions of section 1.3 of chapter 1. There would be no more discussion about the method based on the power dominating set.

2.4.3 Other Graph-Theoretic Methods

Focusing on the methods based on the maximum matching and power dominating set introduced above, an already known state matrix is essential. According to this known state matrix, an input matrix with less columns in number can be identified. Nevertheless, they do not consider the number of non-zero entries of each column of the identified input matrix, which means that each input may not be dedicated for only one system component.

By contrast, in [92], given a state matrix $\mathbf{A}$ of equation (2.1), Pequito *et al.* designed a framework to construct a structurally controllable CT-LTI system with a set of dedicated inputs. It proposes that each input can drive only one internal component, or each column of the identified input matrix has only one non-zero entry, and the number of columns should be minimum. Based on theorem 2.1, given the digraph $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, a method based on a maximum matching and strongly connected components of $G(\mathbf{A})$ is given, while the worst-case execution time is much higher, which is $O(|V_1|^3)$. And the specific method is omitted here, which would be reviewed in section 3.2.2 of next chapter.

2.5 Strongly Structural Controllability

Structural controllability of a CT-LTI system tells us that it is valid to decide the controllability of a CT-LTI system with a (minimum) set of inputs when we do not know the exact values of non-zero entries of both state and input matrices. According to this fact, this section introduces strongly structural controllability (strongly

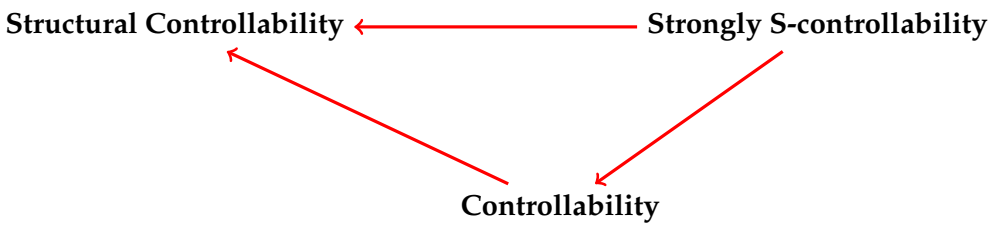

Figure 2.8: Relationships among Basic Control Concepts

In this figure, each red arrow between two control concepts represents that when the concept at its tail is satisfied, it implies the concept at its head.

s-controllability) of a CT-LTI system [80], by which one of our future works are motivated. A system with strong s-controllability is always completely controllable for any exact values of non-zero entries of its both state and input matrices.

Therefore, given a state matrix, to derive controllability of constructing a CT-LTI system, in addition of constructing a structurally controllable system as a way, it is also valid to design a CT-LTI system with the strongly s-controllability. Compared with the definition 2.3 about structural controllability, the strongly structural controllability can be defined below:

Definition 2.19 (Strongly Structural Controllability)
A structured system described by equation (2.1) is strongly structurally controllable, if and only if each instance of it satisfies the controllability rank condition.

For example, in section 2.3.1, the first and the third system, whose graphic representations are **a** and **c** of figure 2.3, are strongly s-controllable systems, because the controllability rank condition is independent with exact values of non-zero entries of both input and state matrices. Also, it is obvious that a strongly s-controllable system is a sufficient but not necessary condition of a structurally controllable system, and a strongly s-controllable system is a sufficient but not necessary condition of a completely controllable system, where such relationship among these three concepts is shown by figure 2.8.

Next, the graph-theoretical condition of a strongly s-controllable CT-LTI system, which was concluded by Chapman $et\ al.$ [31, 32], is shown with definitions from 2.20-2.24 in the following:

Definition 2.20 (Bipartite Graph B_A)
Given a state matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ of equation (2.1), let $B_A = (V_A^+ \cup V_A^-, E_A)$ be a bipartite graph, $V_A^- = \{v_i^-|2 < i \leq n\}$ $V_A^+ = \{v_j^+|2 < j \leq n\}$ be two independent vertex sets, and $E_A = \{(v_j^+, v_i^-)|v_i^- \in V_A^-, v_j^+ \in V_A^+\}$ be an edge set. Besides, let $\beta : \boldsymbol{A} \rightarrow E_A$ be a bijection. For each non-zero $a_{ij} \in \boldsymbol{A}$, there is $\beta : a_{ij} \rightarrow (v_j^+, v_i^-)$.

Definition 2.21 (*Constrained t-matching* [54])
In a non-empty bipartite graph, let M be a matching, and $V(M)^+$, $V(M)^-$ be the vertex sets only containing nodes incident to edges of M, where $|V(M)^+| = |V(M)^-| = t$ $(t \geq 1)$, and $V(M)^+ \cap V(M)^- = \emptyset$. Then, M is a constrained t-matching *(or uniquely restricted matching), if it is the only matching containing t edges between $V(M)^+$ and $V(M)^-$.*

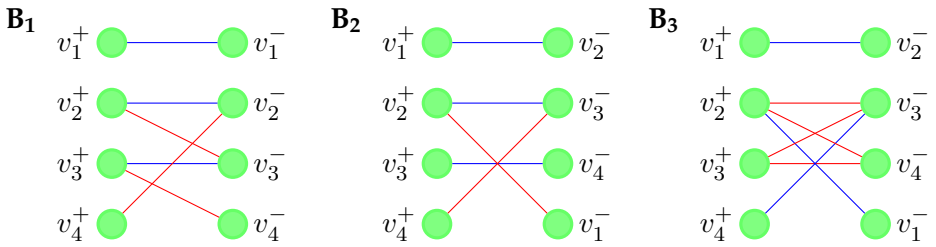

Figure 2.9: Examples of *constrained t-matchings*

The set of all blue edges of each bipartite graph above is a *constrained 3-matching* between vertices incident to those blue edges.

Remark 2 A matching in a bipartite graph is said to be uniquely restricted or a *constrained t-matching* if there is no other matching with the same cardinality as it on the vertices spanned by this matching [54]. (See examples in figure 2.9.) □

Definition 2.22 (Self-less Matching)
*In a non-empty bipartite graph, let M be a matching, and $V(M)^+$, $V(M)^-$ be the vertex sets only containing nodes incident to edges of M, where $|V(M)^+| = |V(M)^-| = t$ $(t \geq 1)$, and $V(M)^+ \cap V(M)^- = \emptyset$. Also, let $V_{sub}(M)^-$ be a vertex set and $V_{sub}(M)^- \subseteq V(M)^-$. Then, M is a $V_{sub}(M)^-$-*less *matching, if it excludes edges like (v_i^-, v_i^+), where $v_i^- \in V_{sub}(M)^-$, and $v_i^+ \in V(M)^+$. If $V_{sub}(M)^- = V(M)^-$, this $V_{sub}(M)^-$-*less *matching M is a* self-less *matching.*

For example, in the bipartite graph $\mathbf{B_1}$ of figure 2.9, the set of blue edges is not a self-less matching for $\{v_1^-, v_2^-, v_3^-\}$ and $\{v_1^+, v_2^+, v_3^+\}$, while the set of red edges is a self-less matching, and it is also a constrained self-less *3-matching* for $\{v_2^-, v_3^-, v_4^-\}$ and $\{v_2^+, v_3^+, v_4^+\}$. Besides, in $\mathbf{B_2}$ and $\mathbf{B_3}$, the sets of blue edges are also constrained self-less *3-matchings* for the sets of vertices incident to those blue edges. According to definition 2.21 and 2.22, a maximum (constrained)(self-less) *t-matching* in a given bipartite graph means that there is no (constrained)(self-less) *s-matching* in this given bipartite graph with $s > t$. For instance, in the bipartite graph $\mathbf{B_2}$ of figure 2.9, $\{(v_1^+, v_2^-), (v_2^+, v_1^-), (v_3^+, v_4^-), (v_4^+, v_3^-)\}$ is a maximum constrained self-less *4-matching*. And in $\mathbf{B_3}$, $\{(v_1^+, v_2^-), (v_2^+, v_1^-), (v_3^+, v_4^-), (v_4^+, v_3^-)\}$ is also a maximum constrained self-less *4-matching*.

Definition 2.23 (Matrix A_X)
Given a state matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ of equation (2.1), let $\boldsymbol{A_X} \in \mathbb{R}^{n\times n}$ be a matrix, and $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ be a matrix, where diagonal entries of $\boldsymbol{I}$ are non-zero and others are all zero. Then, $\boldsymbol{A_X} = \boldsymbol{A} + \boldsymbol{I}$.

Definition 2.24 (Bipartite Graph B_{A_X})
Given a matrix $\boldsymbol{A_X} \in \mathbb{R}^{n\times n}$ of definition 2.23, bipartite graph $B_A = (V_A^+ \cup V_A^-, E_A)$ of definition 2.20, let $B_{A_X} = (V_{A_X}^+ \cup V_{A_X}^-, E_{A_X})$ be a bipartite graph, and $\gamma : \boldsymbol{A_X} \to E_{A_X}$ be a bijection. Then, $V_A^+ = V_{A_X}^+$, $V_A^- = V_{A_X}^-$, $E_{A_X} = E_A \cup \{(v_i^-, v_i^+) | v_i^- \in V_{A_X}^-, v_i^+ \in V_{A_X}^+\}$, and for any non-zero entry a_{ij} or a_{ii} of $\boldsymbol{A_X}$, there is $\gamma : a_{ij} \to (v_j^+, v_i^-)$, or $\gamma : a_{ii} \to (v_i^+, v_i^-)$.

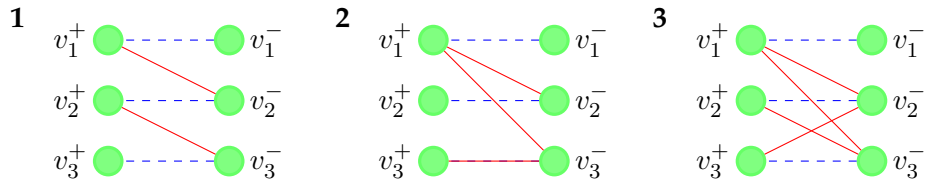

Figure 2.10: Examples of bipartite graphs for strongly s-controllable systems

Given system **a**, **c** and **d** of figure 2.3, three bipartite graphs obtained by definition 2.20 and 2.24, respectively where each bipartite graph above excluding blue dashed lines are obtained by definition 2.20, while blue dashed lines are added into them additionally.

With these definitions, a sufficient and necessary graph-theoretical condition of a strongly s-controllable CT-LTI system is concluded by the following theorem 2.6 and corollary 2.7, which were initially raised by Chapman *et al.* [32, 31]:

Theorem 2.6 (Strongly S-Controllable System Theorem)
Given a system described by equation (2.1), $B_A = (V_A^+ \cup V_A^-, E_A)$ of definition 2.20, and $B_{A_X} = (V_{A_X}^+ \cup V_{A_X}^-, E_{A_X})$ of definition 2.24. Also, let $V_{sub}^- = \{v_i^- | b_{ij} \neq 0, b_{ij} \in \boldsymbol{B}\}$ be a subset of V_A^-, and $|V_{sub}^-| = m$. Then, the given system is strongly s-controllable if and only if there is a constrained (n - m)-matching *in $\{\text{“}B_A\text{”} \setminus V_{sub}^-\}$, and a constrained $\{V_{sub}^-\}$-less* (n - m)-matching *in $\{\text{“}B_{A_X}\text{”} \setminus V_{sub}^-\}$.*

Further, according to theorem 2.6, corollary 2.7 below can confirm a minimum cardinality of a set of inputs to derive a strongly s-controllable system.

Corollary 2.7
*Given a state matrix **A** of equation (2.1), and bipartite graph B_{A_X} of definition 2.24. According to theorem 2.6, consider a maximum constrained self-less* (n - m)-matching *of B_{A_X} with unmatched nodes of V_A^-, where the number of unmatched nodes is m. Then, the minimum number of inputs associated with a strongly s-controllable system containing the state matrix **A** is m.*

Compared with the way to identify a minimum set of inputs to construct a structurally controllable system, identifying a minimum set of inputs for a strongly s-controllable system is a NP-complete problem [31], due to identifying a maximum constrained self-less t-matching [54]. And a greedy algorithm has been developed to just identify a set of inputs of a strongly s-controllable system by Chapman and Mesbahi [31]. Given a state matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, this algorithm is executed in $O(n^2)$ steps at most, while the cardinality of the return is not necessarily minimum.

2.5.1 Examples

Here, we use some systems shown in section 2.3.1 as simple examples to understand the statements about strongly s-controllable systems, which are the system **a**, **c** and **d** of section 2.3.1. The related bipartite graphs are shown in figure 2.10.

1. In the first bipartite graph, based on theorem 2.6 and its notations, $V_{sub}^- = \{v_1^-\}$, $\{(v_1^+, v_2^-), (v_2^-, v_3^+)\}$ is a constrained *2-matching* in the bipartite graph after removing v_1^- and all blue dashed lines, and it is also a $\{v_2^-, v_3^-\} - less$

2-matching in this bipartite graph only after removing v_1^-. Thus, the system represented by **a** of figure 2.3 is strongly s-controllable.

Also, by corollary 2.7, because $\{(v_1^+, v_2^-), (v_2^-, v_3^+)\}$ is still a maximum constrained self-less matching in the first bipartite graph of figure 2.10, the minimum input set is thus a single element set.

2. In the second bipartite graph of figure 2.10, by theorem 2.6, $V_{sub}^- = \{v_1^-\}$, $\{(v_1^+, v_2^-), (v_3^-, v_3^+)\}$ is a constrained *2-matching* in the bipartite graph after removing both v_1^- and all blue dashed lines, and it is also a constrained $\{v_1^-\} - less$ *2-matching* in this bipartite graph only after removing v_1^-. Thus, the system represented by **c** of figure 2.3 is strongly s-controllable. Furthermore, $\{(v_1^+, v_2^-), (v_3^-, v_3^+)\}$ is also a maximum constrained $\{v_1^-\} - less$ matching, the minimum set of inputs is thus one.

3. In the third bipartite graph of figure 2.10, by theorem 2.6, $V_{sub}^- = \{v_1^-\}$. After only removing v_1^-, there are multiple 2-matchings incident to $\{v_2^-, v_3^-\}$: $\{(v_2^+, v_2^-), (v_3^+, v_3^-)\}$, $\{(v_2^+, v_3^-), (v_3^+, v_2^-)\}$, $\{(v_1^+, v_2^-), (v_3^+, v_3^-)\}$, $\{(v_1^+, v_3^-), (v_3^+, v_2^-)\}$, $\{(v_1^+, v_2^-), (v_2^+, v_3^-)\}$, $\{(v_2^+, v_2^-), (v_1^+, v_3^-)\}$.

   Because each of these 2-matchings is not constrained in the bipartite graph after only removing v_1^-, the condition of theorem 2.6 is not satisfied. Thus, the system represented by **d** of figure 2.3 is not strongly s-controllable.

Above all, given a state matrix, to construct a CT-LTI system with controllability, as we can observe from section 2.3 to section 2.5, it can be valid to construct a structurally controllable system according to the maximum matching of a digraph that is mapped by the given state matrix. Besides, it can be also valid to construct a system with strongly s-controllability. In terms of efficiency of identifying an input set, and the minimality of the derived input set in order to derive a completely controllable CT-LTI system, this problem might be more likely to use the maximum-matching based method of section 2.4.1, to obtain the structural controllability of a CT-LIT system with the minimum set of inputs in polynomial time.

2.6 Controllability of Complex Network

Over decades, complex networks influence research in various areas [4, 87, 15, 119]. And this phenomenon also promotes us to further control [73] and observe [50] complex networks and complex systems for more purposes. When a network with CT-LTI dynamics is given, such as the digraph $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, where the state matrix $\mathbf{A}$ of equation (2.1) now shows the interaction among system internal vertices, and the system state vector $x(t)$ now holds the state of each system internal vertices at the moment t. Intuitively, a system including CT-LTI dynamical digraph and the inputs can be also described by equation (2.1), and controllability rank condition can be also used to analyse its controllability.

To completely control a network such as $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4 by a minimum set of inputs, based on above facts about controllability and structurally controllability that are visualized in section 2.3.1, and algorithm 2.1, Liu *et al.* solved this problem through a maximum matching of $G(\mathbf{A})$. And vertices

of $G(\mathbf{A})$ that are directly forced by the inputs are called the driver nodes. Generally speaking, idea of Liu *et al.* is to structurally control $G(\mathbf{A})$, so that a completely controllable system containing $G(\mathbf{A})$ could be obtained in aspect of "statistical physics", which is further generalized by the following theorem:

Theorem 2.8 (Minimum Input Theorem [73])
Given a network, like $G(\boldsymbol{A}) = (V_1, E_1)$ of definition 2.4, the minimum number of inputs, or equivalently, the minimum number of driver nodes to completely control $G(\boldsymbol{A})$ is one, if it contains a perfect matching, where any vertex can be a driver node and directly driven by an input. Otherwise, it is the number of unmatched nodes related to a maximum matching, where only each unmatched node is directly driven by an input.

Currently, because a minimum-input structurally controllable network can be obtained by the maximum-matching method in polynomial time complexity in the worst case, this theorem has been widely used to study controllability of complex networks with LTI dynamics in various areas, such as target drug design [44], interbank networks control [46], epidemic networks [103] and so on. The study of network controllability is to force all network vertices from any initial state to a proposed one by properly applying external inputs in finite steps. And also, with the minimum input theorem [73], research of robustness of network structural controllability against malicious attacks or failures [77, 94] is also further facilitated. Because of the background of statistical physics, the types of studied networks are explicitly argued and the number of edges and nodes are also very big, whose number of nodes or edges might be over tens of thousands, such as Internet. Even though, there exist some networks that are not completely controllable via the identified inputs by the minimum input theorem. For such networks, they are called the pathological cases [104], and further modification is given to make them completely controllable. In [117], adding extra inputs as a way is implemented, and the resulting network is said to be physically controllable. Also, such obtained controllability is called the physical controllability. Nevertheless, related methods to obtain a physically controllable network are not generic, because it is obtained by severe predefined constraints. Also, the constrained-matching method for driving strongly structural controllability introduced in section 2.5, can still be applied to completely control a CT-LTI dynamic network, such as $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4. By contrast, this thesis just uses maximum-matching based method to study the controllability in graph-theoretical aspect, and any minimum set of inputs identified by this method is only for structural controllability.

Additionally, the method based on the power dominating set of a digraph in section 2.4.2, can be also applied to obtain a structurally controllable. Because of this, robustness of network structural controllability against malicious attacks for different kinds of graphs is also studied [6]. Nevertheless, after this chapter, the method based on the power dominating set to derive structural controllability would be not considered further.

Chapter 3

Related Works

3.1 Overview

According to research questions of section 1.3.3 of chapter 1, this chapter firstly shows recent works about structural-control recovery. Then, this chapter reviews related works about robustness of network structural controllability against both nodal and edge removals in various scenarios. Additionally, related works that analyse network vertices and arcs to maintain structural controllability with a minimum set of inputs are also reviewed as well. In particular, some graph-theoretical problems are also reviewed, which are used to model and solve problems such as structural controllability recovery and network analysis with different constraints.

3.2 Structural-Controllability Recovery

Structural-controllability recovery is to render a currently structurally uncontrollable system be structurally controllable again, which means the recovered system was structurally controllable prior to the attack or failure. Generally speaking, it needs to obtain an input matrix according to the given state matrix, so that the final system is structurally controllable, and it is desirable to complete such identification as soon as possible rather than recomputing an input matrix. Nevertheless, in special, it is also possible to recover structural controllability based on the given input and state matrices together.

3.2.1 Recovery without Input Constraints

This section mainly shows previous works related to research question 1 and 2. According to chapter 2, structural controllability could be acquired by several ways. One is by the maximum-matching based method of section 2.4.1. After malicious attack or random failure on network vertices, Ding $et\ al.$ [48] recovered network structural controllability with a minimum set of inputs by optimally adding extra input nodes to the residual digraph. However, in their recovery scenario, any assumptions of the input digraph are not given. After removing some nodes, they identified a maximum matching to understand how many unmatched nodes should be forced. Except for those currently identified unmatched nodes that are also now forced by previously inputs, remaining ones are arranged new input nodes by the minimum input theorem 2.8. Nevertheless, this scenario does not increase any efficiency compared with the recomputation. Besides, another structural-controllability revovery method is based on the power dominating set [59], which is introduced in section 2.4.2. Again, by identifying a minimum power dominating

set, after removing vertices, Alwasel $et\ al.$ [12] recovered network structural controllability of *Erdős-Rényi*(ER) random digraph in LTI model. As we already know, identifying a minimum power dominiating set of a digraph is a NP-complete problem. Thus, to enhance the efficiency of recovery, Alwasel $et\ al.$ assumed that the input digraph is a weakly connected digraph, excludes self loops, parallel edges and isolated nodes. In addition to, they also assumed that the input digraph can be decomposed into trees with bounded tree width. In [9], they reconstructed a minimal power dominating set of a resulting graph with bounded tree width according to the original work of Aazami, Stilp [1] and Guo [56]. As a result, the worst-case time complexity is $O(nc^k)$, average-case time complexity is $O(log(nc^k))$, where n represents the number of nodes of a given graph, c is a constant number, and k is the bounded tree width. Then, based on the subgraph of the input digraph, which is identified by using the depth-first search [111], Alwasel $et\ al.$ proposed a novel power dominating set algorithm [10] by re-using the left known power dominating set of previously structurally controlled network. Compared with the previous result of [9], this novel algorithm based on DFS improves the average-case complexity [10], while the worst-case time complexity is not changed. The average-case time complexity is represented: $O(|V \cup \{E_t \cup E_f[N(v)] \cup E_c[N(v)]\} \setminus E_b|)$, where the single vertex v belongs to the identified power dominating set, E_f, E_c and E_b are three different types of edges of input network, $E_c[N(v)]$ represents a set of neighbours of v in E_c, and V is the vertex set of input network. For exact definition of these three kinds of edges, please refer to that paper. After that, they [11] gave an approximation of an efficiently reconstructed power dominating set via a block decomposition on the input digraph. In this work, each removed single node is specially defined, and recovery is executed based on each removed node. The worst-case time complexity of recovery after removing a set of nodes is $O(nc^W)$, where W is the number of neighbours of a single removed node, and c is a constant number. Besides, Alcaraz $et\ al.$ [7] relies on the minimum power dominating set to recover structural controllability of general power-law and scale-free digraphs in CT-LTI model, which also forbids recomputing a new power dominating set. Again, those given digraphs have no self loops and must be acyclic and structurally controllable by a known power dominating set in the beginning. Two main repair approaches are raised against deletion of already known nodes and edges. In detail, one approach is to connect unobserved vertices with remaining nodes of previously known power dominating set. The other one is to use a previously backup power dominating set that can observe nodes that are unobserved by the currently remaining power dominating set. These two strategies are executed in $O(|U||V|^2)$, in which U is the set of nodes out of remaining dominating set and V is the set of all vertices.

In conclusion, after malicious attacks or random failures on system vertices or components, structural control recovery can be effectively executed via several ways in polynomial time. However, these existing structural controllability recovery methods did not consider the amount of attacks or failures. If those methods above are used to recover structural controllability after very limited modification, unnecessary computation might be paid. For such a situation, without recomputing a new maximum matching and based on previously identified minimum set of inputs, chapter 4 and 5 recovers structural controllability with a minimum set of

inputs after a single vertex addition and removal, respectively.

Because of the maximum-matching based method to acquire structural control, after a single network vertex or system component modification, structural control recovery is thus further modelled into the dynamic graph problem [124]. Such a graph theory problem seeks to efficiently maintain a data structure after a change rather than recomputing this data structure. A fully dynamic graph problem addresses the update operations of unlimited insertions and deletions of edges or vertices, while a partially dynamic only considers either insertions or deletions of edges or vertices. Fully dynamic approximate maximum-cardinality matching problem is popular in recent years. In the following, m and n represents the number of edges and nodes of a given network. In 2010, Rubinfeld $et\ al.$ [90] designed a randomized algorithm that maintains a $O(1)$-approximation maximum matching in $O(log^2 n)$ time. Baswana, Gupta and Sen [18] then gave a 2-approximation maximum matching in a dynamic graph with $O(log\ n)$ amortized time. In [21], with a deterministic data structure, the approximation ratio is $(3/2+\epsilon)$ and the worst case time complexity is $O(m^{1/4}\epsilon^{-2.5})(\epsilon > 0)$. Until now, [23] presented a deterministic data structure with $(2+\epsilon)$-approximation and the worst-case time complexity is $O(log^3 n)$. Nevertheless, to derive an exact size of a maximum matching in the fully dynamic, the best known update time is $O(n^{1.495})$ [102]. In comparison, research question 1 solved in chapter 4 is partially dynamic and each update only allows adding a single node. Besides, algorithms of chapter 4 are deterministic and only concern the worst-case complexity. On the other hand, because the removed single vertex is known, structural-controllability recovery in chapter 5 that addresses question 2 can not be modelled into a dynamic graph problem.

3.2.2 Recovery with Input Constraints

Nevertheless, related works of section 3.2.1 neglect constraints on inputs during structural-control recovery. By contrast, this section shows previous works related to research question 3. It is more realistic and sufficient to concern constraints on inputs during the process of recovering structural controllability. Constraint on inputs might be on the number of inputs, or the adjacency between inputs and system components. When such requirements are concerned, recovery might require extra modification, such as adding extra non-zero entries into the given state matrix. And it is more likely to recover structural controllability according to original graph-theoretic conditions of theorem 2.1 of chapter 2, rather than the minimum input . In chapter 6, this problem is solved by a minimum-edge addition. As a related work for this chapter, in [37], Chen $et\ al.$ proposed to get structural controllability by the minimal edge addition. Their edge-addition scenario is mainly based on the work of [91] and [92], which obtains the structural controllability with dedicated inputs by theorem 2.1 of chapter 2. Specifically, they proposed to confirm a set of disjoint cacti of definition 2.8 within the system network of a given system, where each input can only force a single system component. And the set of inputs should be minimum. For their solution, since strongly connected components [42] of the system network contains all cycles of any identified maximum matching of this network, the strongly connected components identified by a maixmum matching are used. From directed paths of an arbitrarily identified maximum matching, they eliminated inaccessible strongly connected componenets that contains disjoint

cycles, and further confirmed vertices directly forced by dedicated inputs. Those nodes could be a single node of each inaccessible components, or starting nodes of those directed paths. Nevertheless, this method directly considers all strongly connected components, and also uses the Hungarian algorithm [68] to reduce the number of added edges. As a result, the worst-case execution time is proportional to the cubic number of vertices of the given system network. In comparison, the minimum-edge addition scenario of chapter 6 that addresses question 3 is more efficient, whose time complexity is equivalent to identifying a maximum matching.

3.3 Robustness of Network Structural Controllability

Robustness is the ability to withstand failures and perturbations. It is a critical attribute of complex systems and networks [116]. A fundamental issue concerning the functioning of a complex network is the robustness of the overall system to against the failure on its constituent parts [5]. Therefore, intuitively, robustness of network structural controllability is the ability to keep structural control into the residual network after malicious attacks or random failures. Robustness of structural controllability has been widely studied, especially after raising the minimum input theorem 2.8 of chapter 2. In this section, previous works related to robustness of network structural controllability against nodal and edge removals are shown, where the structurally controllability is with a minimum set of inputs. Conclusions obtained from those works generate the general motivation for the part III of this thesis, which concentrates on the efficient network analysis to identify vulnerable single nodes and edges to the removal.

3.3.1 Robustness Against Node Removals

According to the minimum input theorem, robustness of network structural controllability is studied for different kinds of networks against various network-vertex removal scenarios, which is qualitatively measured by the fluctuation of the minimum number of inputs that structurally control the current residual network. Although there are other measures to quantify robustness against node removals, the minimum number of inputs is always used as a critical parameter for other measures [121, 115].

By the simulation results obtained in [94], given the scale-free (SF) [25] and ER random [51] digraphs, each of which contains 1000 nodes, and with the average degree is 6. The power law parameter is $\gamma_{in} = \gamma_{out} = 3$. Then, given a set of ordered time steps, in each step, a network node is randomly removed, while the degree-based attack removes a node with the largest degree. After removing a node, both scenarios calculate the minimum number of inputs to structurally control the latest residual network. For both types of networks, with the increase of the number of removed single vertices, the minimum number of inputs to structurally control the current remaining network is dramatically rised. Also, degree-based attacks are more harmful to network structural controllability than random single-node removal [94] for SF and ER random digraphs. Similarly, in [77], given the ER random digraph, small world digraph, each of which includes 200 nodes, 400 edges, and with average degree 4. And given a scale-free digraph with 200 nodes, 591 arcs and

average degree 5.91, it also obtained a same conclusion for the same single-node removal scenario as [94], which means that continuously removing single vertices harms the minimum-input structural control into both large and small networks. Further, the degree-based attack was studied through the initially calculated degree of the original network, and the recalculated one of each current network, respectively. Based on the order of that originally calculated or every recalculated vertex degree, each removed node is determined. And they also considered the vertex betweenness attack. Given a vertex, its betweenness is the number of shortest paths through it, and the removed vertex's betweenness is either initially calculated or recalculated. The numerical results of [77] illustrate that structural controllability with a minimum set of inputs of their ER random networks are more robust than scale-free networks against a same node-based attack.

In addition to single vertex removals, authors of [94] and [109] focused on cascading failures triggered by the removal of the network vertex that has the largest load [53] [84], where the maximum load of each node is defined as its betweenness. The removal of a node with the largest load may increase the load of some other vertice, which might be larger than their capacity, where the overloaded nodes fail and all their connections with fail nodes are removed as well. The failures of these overloaded nodes result in a new distribution of load on remaining nodes, on which some nodes may fail in the existing network for the same reason. The cascading failures continue until there are no overloaded nodes in the residual network. The results show that cascading failures on the vertex is more harmful to structural controllability of directed scale-free and ER network than single-node removals. Besides, from the results of simulation of [109], given networks of 1000 nodes, robustness of ER and scale-free digraphs is proportional to average degree that belongs to $[0, 18]$, or probability of ER graphs that belongs to $[0.05, 0.2]$, and power-law parameter of scale-free graphs that belongs $[2.2, 4.0]$.

On the other hand, according to the power dominating set [59], in [6], authors studied the robustness of structural controllability of networks with ER random model, scale-free model and small-world model [119]. Different from those previous works, they did not use the change of the minimum number of inputs to measure related robustness. Rather, the diameter, density, and average clustering coefficient of the residual network after removing a vertex are concerned to measure the robustness of network structural controllability against nodal removals. Here, the driver node set is derived by a power dominating set, which is identified through maximum out-degree vertices, minimum out-degree nodes and in random, respectively. Also, attack model is to remove a single node, based on the order of the given driver node set, or vertex betweenness, or in random. From their numerical simulation results, roughly speaking, for the network with the number of nodes less than 500, those items would be affected heavily, while for networks with more vertices, they perform stably during the attack.

3.3.2 Robustness Against Edge Removals

The minimum number of inputs for each residual network is also used to measure the robustness of network structural controllability against single-edge removels. In [100], given some ER random digraphs, whose number of nodew varies from 200 to 300, and the number of edges is certain, which is 2000. Then, during the

process of removing single edges from the original network, the minimum number of inputs to structurally control the latest remaining network maintains stably for a while. After a certain fraction of removed edges on the original number of edges, that minimum number of inputs surges rapidly. Notwithstanding, in [77], for random removal of single edges one by one from the given ER and scale-free networks that contain hundreds of nodes and edges, the minimum numer of inputs continuously increases along with increasing fraction of removed single edges on the original total amount. Besides, it is also concluded by [77] that recalculated edge betweenness attack are more harmful than edge-degree based attack and initially calculated degree and betweenness attacks for those given ER random digraphs, which is reflected by the change on the amount of inputs to structurally control current residual network. Simultaneously, with ER random networks, small world networks with 200 nodes and 400 edges, and scale-free networks with 200 nodes and 591 edges, authors of [77] further observed that the node-based strategies are often more harmful to the network structural controllability than the edge-based ones, and so are the recalculated strategies than their counterparts.

Additionally, Nie *et al.* [88] explored the robustness of network structural controllability against the cascading edge failure. They defined that the maximum load of an edge is the total number of shortest paths passing it. Then, the first attack model is the random removal of a set of edges; the second one is the intentional removal of a set of edges in descending order of the initial edge load. From their numerical results, given ER random directed networks of 10000 nodes, each of whose average degree is 2, 4, 6, 8, respectively, removing the highest load edge triggers cascading failures easier in the network with lower average degree, which is lower than or equal to 6. Further, it is observed that the higher the average degree is, the more difficult to trigger cascading failure by removing the highest load edge, where the average degree is 10 or larger. Besides, for small average degree ER networks, which is 2, both the random and intentional attacks cannot cause the cascading failures in those given ER networks. However, for large average degree ER networks, the minimum number of inputs decreases initially for intentional attacks, then, it surges. By contrast, for the same large average-degree ER networks, which is 8, randomly removing edges slowly increases the minimum number of inputs and decreases then. After a certain point of edge removal amount, it dramatically increases a lot.

In summary, it seems that global properties of a network, such as vertex average degree, betweenness of single vertex and edge, are more likely used to investigate robustness of network structural controllability against vertex and edge removals. However, from these results of different vertex and edge removal scenarios, it is still unknown how each removed single node or edge quantitatively determines the minimum set of inputs to structurally control the residual network. As one of possible results, vulnerable single vertices to the removal can not be explicitly identified, and let along protecting network structural controllability with a minimum set of inputs. Also, effectively estimating robustness of network structural controllability against a kind of removal scenarios, and quantitatively explaining the fluctuation of the minimum number of inputs is also essential during the period of removing vertices or edges. For these desirable works, they would be done in following chapter 7 that addresses question 4 and 9 that addresses question 6.

3.4 Network Analysis for Structural Controllability

In general, due to the minimum set of inputs or driver nodes are not unique, what the role of an individual node or edge plays in structural control motivates people to analyse and classify network vertices and edges. Following previous works are all about such related analysis. Particularly, all of them relies on the maximum-matching based method, or the minimum input theorem to derive structural controllability with a minimum set of inputs.

3.4.1 Edge Based Analysis

This section shows previous works related to the research question 4. To clarify the importance of an edge in maintaining network structural controllability, Liu $et\ al.$ [73] raised critical, redundant, and ordinary categories: a removal of a critical edge gains the minimum number of inputs to structurally control residual network; removing a redundant edge never affects currently minimal inputs; removing an ordinary link changes the control configuration, except for the minimum number of inputs. Liu $et\ al.$ claimed that such those three kinds of edges can be effectively identified by the algorithm of [97]. In detail, to confirm the category of a given single edge, this algorithm iteratively identifies a maximum matching of a bipartite graph mapped by the given digraph after removing this edge. To confirm category of each edge, recomputing a maximum matching would be executed until all edges are removed once. As a result, given a digraph with n vertices and m edge, time complexity of this algorithm is $O(m^2 \cdot \sqrt{n} + \sqrt{n} \cdot m)$ in the worst case. Obviously, such algorithm might be infeasible for large-scale networks analysis. Therefore, we are motivated to efficiently classify all edges of a given network into those categories in chapter 7, whose worst-case execution time is $O(m + n + \sqrt{n} \cdot m)$.

The problem of edge classification [2] always attracts the attention of various research areas, especially in artificial intelligence and data mining over years. Generally, given a graph $G = (V, E)$ (a social network mostly), where V and E are vertex set and edge set, a subset $E_0 \subseteq E$ has been labelled or classified in advance. Then, edge classification problem is raised to determine the labels on categories of edges of $\{E - E_0\}$. Chronologically, this problem was initially formalized by Liben-Nowell $et\ al.$ [71], called the link-predition problem, on which people proposed to predict new interaction among existing nodes in a social network by analysing proximity among nodes. And some other recent related works can be found in [70], [39], [122] and [3]. Yet, it is very seldom to see that there exists the secure-aware edge classification, let along to protect the network structural controllability against attack or failure on edges.

Although there have been three lables: critical, redundant and ordinary, defined by Liu $et\ al.$ [73], there is no previously labelled edges of the given network, and it is needless to predict the label of each edge. Thus, solution to general edge classification problems is not suitable for our this classification. Rather, chapter 7 that addresses question 4, accurately and efficiently confirms edges of each category by searching each single edge involved into a maximum matching of an input network, each of which is called the maximally-matchable edge with respect to a given maximum matching.

Searching maximally-matchable edges of a general graph has been pervasively studied over recent decades. Generally, given a maixmum matching, any edge out of it is said to be maximally-matchable with respect to this maximum matching if and only if it can construct a different maximum matching by replacing an edge of it. Initially, Rabin and Vazirani [95] designed a randomized algorithm to find all maximally-matchable edges in general graphs that contain a perfect matching with time complexity of $O(n^{2.376})$, where n is the nummber of graph nodes. Then, still with general graphs, [38] gave a distinct randomized algorithm to find the Gallai–Edmonds decomposition, which is also a way to find maximally-matchable edges in polynominal time of $O(n^{2.38})$. For deterministic algorithms, with the same graph and purpose, Carvalho and Cheriyan [30] found edges in at least one perfect matching, called ear decomposition of a matching-covered graph. Their deterministic algorithm runs in $O(nm)$ steps at most, and m represents the number of edges. Besides, Costa *et al.* [43] solved problems about finding maximally-matchable edges in a bipartite graph. They identified edges of a bipartite graph into three partitions: E_1 whose edges belonging to all maximum matchings; E_0 whose edges out of any maximum matching; edges involved into E_w is neither in E_1 nor E_0. By finding E_1 and E_w, all maximally-matchable edges are obtained, and the time complexity is $O(nm)$. Compared with the worst-case execution time, Tassa [112] claimed that the worst-case execution time of finding all maximally-matchable edges in a bipartite graph with a known maximum matching is reduced to $O(n+m)$ time. He classified all maximally-matchable edges into few categories. Reviewing his method, we found a problem. In detail, Tassa applied the breath-first search(BFS) [42] to find some arcs in a digraph, which is mapped by the input bipartite graph, as a way to find some kinds of maximally-matchable edges. However, the BFS algorithm might not traverse all arcs of a digraph except for tree digraphs, it means that some arcs corresponding to valid maximally-matchable edges of the input bipartite graph may be missed. As a result, Tassa's method can not always find all maximally-matchable edges in a bipartite graph with a known maximum matching. By contrast, algorithms of chapter 7 are all deterministic and only concern the worst case execution, where all maximally-matchable edges of an input network can be identified in linear time except for precomputing the known maximum matching of the input network.

3.4.2 Driver-Node based Analysis

According to the description of section 2.6 of chapter 2, the driver node is significant in deriving structural control into CT-LTI dynamical networks, and driver nodes might be easily targeted by malicious attacks, such as control hijack [34], where attackers influence a set of identified driver nodes by the maximum-matching based method of section 2.4.1. Thus, this section shows previous works related to the research question 5, which concentrates on how a network vertex can become a node of a minimum set of driver nodes. And such identification can be used to protect driver nodes against malicious attacks and failures in advance.

Jia *et al.* [62] firstly classified a vertex into critical, redundant or intermittent categories, if it is always, never or sometimes, included by a mininum set of driver nodes to control a given network. Based on this network-vertex classification, all vertices able to be involved into a minimum set of driver nodes are identified with

time complexity of $O(N \cdot L)$ in the worst case, where N, L are the number of nodes and arcs of a given digraph. In detail, they proved that critical node is the vertex without indegrees, which can be identified in linear time. Then, any intermittent node can be confirmed by removing it and identifying an augmenting path related to current matching, which costs $O(N \cdot L)$ steps at most. If so, this node is intermittent, otherwise, it is a redundant nodes. Obviously, critical nodes are the most vulnerable vertices to control hijack. Even though, critical nodes might be identified and protected in advance, while intermittent nodes may thus become new targets of attackers. For this reason, it is desirable to more efficiently identify each node of all minimum sets of driver nodes than the method of [62]. Fortunately, chapter 8 can do it with time complexity $O(\sqrt{N} \cdot L)$ at most. Meanwhile, Ruths $et\ al.$ [101] concluded that any driver node set must contain the nodes without either indegree or out degree. And nodes having both indegree and outdegree can also be a driver node. But there was not a method to identify all nodes able to be a driver node. Recently, Peter, and Cohen $et\ al.$ [40] analysed driver nodes among randomized networks, where the randomization does not change the initial graph degree distribution. They find that some nodes are always driver nodes during the given network under randomization. In [131], authors claimed that they can find all nodes able to be driver nodes in linear time. Nonetheless, they did not prove that the number of those found vertices are maximum, so that if the found nodes are all nodes of all minimum sets of driver nodes is unknown. Similar to the classification of Jia $et\ al.$ [62], Commault $et\ al.$ [41] classified nodes for network structural controllability, while each external input is stimulated to be dedicated, which means that each input can be adjacent to only one node.

Additionally, since single-vertex removals can dramatically increase the minimum set of driver nodes in cardianlity [94] to control the residual network, and those related works above are all unable to show possible impacts of removing any single driver node on controlling the residual network with a minimum set of driver nodes. Therefore, after finding each node involved into all minimum sets of driver nodes, chapter 8 that solves question 5 is also motivated to explore impacts of a single driver-node removal on the minimum set of driver nodes to structurally control the residual network.

3.4.3 Generic-Vertex Based Analysis

Reviewing literatures of section 3.3, it is still unknown how each removed node determines the minimum set of inputs to structurally control the residual network, so that vulnerable single vertices to removals can not be explicitly identified, let along protecting network structural controllability with the minimum set of inputs against single-node removals. For this reason, it is also urgent to clarify impacts of an arbitrary single-node removal on the minimum set of inputs to structurally control the residual network. Then, this section illustrates previous works, which are also related to the research question 6.

In recent years, various indices or categories for the single vertice have been created to show the significance of a single node in obtaining controllability of networks. Strictly speaking, here, controllability with a minimum set of inputs is actually derived via the structural controllability by the maximum-matching based method of section 2.4.1 of chapter 2. Chronologically, Liu $et\ al.$ [74] introduced

control centrailty to capture the dimension of structurally controllable subspace through any given single node. Quantitatively, control centrality of a single vertex is equal to the maximum number of vertices that are approached by it via existing directed paths. By the method of [93], such subspace can be identified in linear time. Meanwhile, Wang *et al.* [114] also raised a slightly similar single-vertex index, called control range. Control range of any single node quantifies the number of nodes approachable through directed paths by it, which are also matched nodes related to a same maximum matching. However, since there may be multiple submaximum matchings with a same unmatched node in a network, control range of a same node might have several values. As a solution, control range of a single node is approximated by a small number of submaximum matchings, where some vertices that are always contained by maximum matchings are excluded by those submaximum matchings. In other aspect, Jia and Barabasi [61] introduced control capacity to quantify the probability of a single node to be directly forced by an input in controlling the whole network, which is the fraction of the number of control configurations that involve this node, on the total number of control configurations [61], where the control configuration of a given network is equivalent to an identified maximum matching of a same network. Although all control configurations can be obtained by enumerating all maximum matchings of the given network, the number of maximum matchings grows exponentially with network size [123], and precisely calculating control capacity of a vertex might be computationally prohibitive and infeasible. They thus created a random samplying algorithm to calculate control capacity of a single node. Also, similar to control capcaity, Ding and Lu [47] used control backbone to explore the effort of a single node in controlling the whole network. Control backbone of a single node is the frequency of this node in all minimal control schemes of the network, where the minimal control scheme is a minimum set of driver nodes virtually. Hence, for computational efficiency, Ding *et al.*, designed sampling algorithms to calculate control backbone of each vertex in order to avoid enumerating all maximum matchings. Additionally, without enumerating maximum matchings of a digraph, Commault *et al.* [41] classified all nodes into three categories that are resembling critical, intermediate and redundant categories defined by Jia *et al.* [62] in polynomial time, while each input can be adjacent to only one vertex of the given network.

Notwithstanding, we find two problems about above single-vertex indices. One is massive calculation of those nodal indices of [47, 61] in practice. The other problem is non-indication of nodal importance. The importance here is about how a single vertex maintains the control into a network with a minimum set of inputs. For example, if the value of control backbone or control capacity [61] for a single node is one, this node looks very important to approach control. In spite of this result, if this node has no any out degrees, its removal may be benefical to reducing the minimum number of inputs and ease the control into entire network. On the other hand, if the control centrality [74] or control range [114] of a single node is the number of all vertices of the network, this node might be indispensable to control other nodes. However, if this node is the starting vertex of a directed path, its removal would not increase any difficulty to control the residual network. Additionally, in [113], authors classified a single node into indispensable, dispensable or neutral if its absence increases, reduces or does not change the minimum number of

driver nodes to structurally control the residual network, while there is no further quantitative description and method about classifying entire network nodes. And the only method classifying a single node mentioned in [113] is by recalculating the minimum number of driver nodes of the residual network.

After reviewing above related works, chapter 9 that solves question 6 is highly motivated to define new nodal categories to concisely represent the importance of each involved node in maintaining network structural controllability with a minimum set of inputs at the beginning. By the maximum-matching based method, the impact of any single node removal on the cardinality of a digraph's maximum matching is concerned to conclude our nodal categories. Accordingly, by precise deduction, chapter 9 concludes that a node is critical, redundant or ordinary if its removal increases, reduces, or keeps the initial number of unmatched nodes related to the known maximum matching.

To classify vertices of a given network into few categories, it is also necessary to review existing approaches of the node classification problem. Generally, given few predefined labels or categories of vertices, and a graph involving a set of previously labeled nodes, the node classification problem is raised to infer labels on all currently unalbeled nodes [22]. Solutions are mainly based on the local classifier method or the random walk method. The local classifier method [85] uses defined features of edges incident to those initially labelled nodes to construct a classifier at the beginning. Then, iteratively applying that classifier to determine labels on existing unlabelled nodes. Besides, the random-walk based method [13] determines the most likely label as the final label on a given node by the probability of a random walk starting from this node and ending at an already labelled node.

Because our classifier is based on all numerical impacts of a single node removal on a digraph's maximum matching, and there are no already labelled nodes in the beginning. Neither local classifier nor random walk based methods can precisely classify each node into one of predefined categories rather than by inference to illustrate what a role of a node plays in maintaining the structural control with a minimum set of inputs. By the idea of [113], given a single vertex, after removing it from the network, recomputing a maximum matching of a digraph can conclude its category. Specifically, let m, n be the number of edges and nodes of a digraph, since the category of any node can be confirmed by computing a maximum matching in the residual network, the time complexity of classifying all vertices would be $O(n^{1.5} \cdot m)$. For identifying all vulnerable vertices to removals, such method is not applicable in practice. By contrast, chapter 9 would show a process of entire node classification able to be done in just $O(n + m + \sqrt{n} \cdot m)$ steps at most.

Part II

Efficient Structural-Controllability Recovery

Chapter 4

Iterative Recovery of Structural Control by the Maximum Matching

4.1 Overview

As mentioned in section 1.2.6 of chapter 1 and section 3.2.1 of chapter 3, related works about recovery of structural controllability did not consider the amount of modification on the given CT-LIT dynamical system or network. In consequence, applying those recovery scenarios in practice, might lead unnecessary computation, when the modification is very limited. Although such modification is very limited, it could emerge periodically, and iterative recovery of structural controllability might be indispensable, sometimes. For instance, in the study of robustness of network structural controllability against continuous single-vertex removals [115] [100] [88], and the study of optimizing structural controllability [118, 120, 121], after each network modification, a minimum set of inputs is computed to structural control the resulting network. Particularly, each single recovery is still proposed to be done as quick as possible after the latest modification. So that the entire iterative recovery would be executed more effective, or with lower time complexity.

Therefore, to recover network structural controllability after a very limited modification on network vertices, and in order to increase efficiency of iterative recovery of structural controllability, this chapter solves research question 1. Specifically, with assumptions of section 1.3.2 of chapter 1, given a CT-LTI dynamical network with a precomputed minimum set of inputs as an input network, this chapter seeks to efficiently recover structural controllability of this input network after adding a single vertex. In addition, it is assumed that this input network already contains an pre-identified maximum matching, so that it is initially structurally controllable by a minimum set of inputs.

According to theorem 2.8 of chapter 2, this question is solved by identifying a maximum matching of the resulting network without recomputation. Further, with the pre-identified maximum matching, by corollary 2.4 and definition 2.12 of chapter 2, it is clear and indispensable to find augmenting paths related to it. During the execution, a bipartite graph that mapped by both input network and added single vertex and arcs, is used to ease augmenting-path identification. The problem is eventually solved by identifying a maximum matching of this bipartite graph without recomputation. As a result, in the worst case, with those assumptions above, the maximum matching of the resulting network can be identified in linear time except for computing the minimum set of inputs to structurally control the input network.

For the contribution, given any initially minimum-input structurally controllable network with CT-LTI dynamics, structural controllability with a minimum

set of inputs of it after adding a single vertex can be efficiently recovered in linear time in the worst case, and original inputs are reused with maximum number. This result further reflects that structural-controllability recovery should concern the amount of change on the initial network.

Remaining chapter is structured as follows: section 4.2 defines fundamental graphs and formulates research question; section 4.3 gives solution, and the last section 4.4 summarizes this chapter.

4.2 Problem Formulation

This section firstly defines the input network of the research question. Then, a bipartite graph that is mapped by the input network is defined later. With them, the research question is clearly specified.

4.2.1 Fundamental Graphs

Definition 4.1 (Input Network of chapter 4)
Let $D = (V, E)$ be a large, finite digraph, and D excludes self loops, parallel arcs and isolated nodes. Also, V represents the vertex set, $V = \{v_i | 1 \leq i \leq n\}(n > 1)$, and E represent the edge set, $E = \{\overrightarrow{\langle v_i, v_j \rangle} | v_i, v_j \in V\}$. Besides, let M_D be a fixed and arbitrary maximum matching of D, identified by the algorithm of [60].

With $D = (V, E)$, a bipartite graph is obtained, which is defined by following definition 4.2:

Definition 4.2
Given $D = (V, E)$ and M_D of definition 4.1, let $B = (V_B, E_B)$ be a bipartite graph, V_B^+ and V_B^- be two independent vertex sets, and M_B be a maximum matching of B, where $|V_B| \leq 2|V|$, $|E_B| = |E|$, and $|M_D| = |M_B|$. Then, $V_B = V_B^- \cup V_B^+$, where $V_B^+ = \{v_i^+ | 1 \leq i \leq n\}$, $V_B^- = \{v_j^- | 1 \leq j \leq n\}$, and $V_B^- \cap V_B^+ = \emptyset$. Besides, let $\alpha : E \to E_B$ be a bijection, for each $\overrightarrow{\langle v_i, v_j \rangle} \in E$, there is $\alpha : \overrightarrow{\langle v_i, v_j \rangle} \to (v_i^+, v_j^-)$, where $(v_i^+, v_j^-) \in E_B$. Also, let M_B be mapped from M_D.

An example of the bijection is shown by figure 4.1. And there can not be the edge like $(v_i^-, v_i^+) \in E_B$, because D excludes self loops.

4.2.2 Problem Formulation

By the mapping between a state matrix and a graph of definition 2.4 of chapter 2, it is assumed that $D = (V, E)$ of definition 4.1 and a minimum set of inputs construct a structurally controllable system, which is represented by a state equation below:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{4.1}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is mapped by D, each arc of E only corresponds a non-zero entry of $\mathbf{A}$, and the number of columns of matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ is minimum. Then, let $\mathbf{A}' \in \mathbb{R}^{(n+1)\times(n+1)}$ be a state matrix, which is obtained by adding one row and one column, both of which has at least one non-zero entry into matrix $\mathbf{A}$. Besides, let $\mathbf{B}'$ be an input matrix. Above all, our research question is defined:

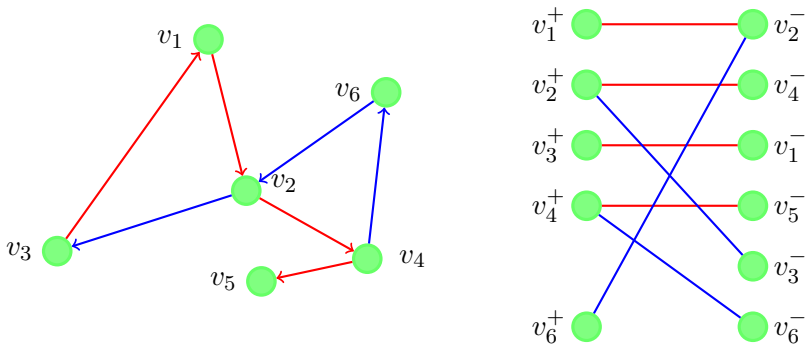

Figure 4.1: An example of bijections of definition 4.2.

A digraph contains a maximum matching that is a red path, which is mapped into a bipartite graph by definition 4.2 with a maximum matching that is all red arcs.

Research Question: Given **A**, **B** and **A**$'$. Then, efficiently identify an input matrix **B**$'$ with a minimum number of columns, so that the dynamical system described by equation:

$$\dot{x}(t) = \mathbf{A}'x(t) + \mathbf{B}'u(t) \tag{4.2}$$

is structurally controllable.

From the graph-theoretical aspect, let u be an added single vertex into $D = (V, E)$ of definition 4.1, and let E_u be a finite set of edges incident to u, where $|E_u| \ll |E|$. Then, according to theorem 2.8 of chapter 2, this research question can be transferred into following problem:

Problem 1: Given $D = (V, E)$ with M_D and $\{u, E_u\}$. Then, recover structural controllability of digraph $(V \cup \{u\}, E \cup E_u)$ by identifying a maximum matching of it without recomputation.

Given $(V \cup u, E \cup E_u)$, and $B = (V_B, E_B)$ with M_B and the bijection of definition 4.2, let E_{B_u} be an edge set mapped by E_u and $|E_{B_u}| = |E_u|$, so that there is $\alpha : E \cup E_u \to E_B \cup E_{B_u}$. Also, let u^+, u^- be nodes incident to edges mapped by arcs incident to u, where $u^- \notin \emptyset$ and $u^+ \notin \emptyset$ depends on if u is the head or tail of arcs of E_u. Besides, it is also assumed that $|E_{B_u}| \ll |E_B|$. Then, **Problem 1** is further transferred into the following problem:

Problem 2: Identify a maximum matching of the bipartite graph $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ with a known matching M_B, rather than recomputation.

Problem 4.2.2 is solved in following section as a way to address **Problem 1** and the research question from graph-theoretical aspect.

4.3 Solution

4.3.1 Maximum matching Identification

By corollary 2.3 and corollary 2.4 of chapter 2, given $B = (V_B, E_B)$ with M_B of definition 4.2, to identify a maximum matching of bipartite graph $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ of problem 4.2.2, it is essential to find augmenting paths related to each latest-identified matching according to the only known matching M_B.

In the first place, for the purpose of avoiding unnecessary computation to identify augmenting paths, following theorem 4.1 clarifies the existence of augmenting paths related to each latest-identified matching based on M_B.

Theorem 4.1
Given bipartite graph $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ of problem 4.2.2 and matching M_B of definition 4.2. Then, two terminals of any single augmenting path related to the latest-identified matching through M_B can not be two unmatched nodes of V_B at the same time.

PROOF Focusing on the latest-identified matching based on M_B, correctness of this theorem is proved through following cases, which are raised by the cardinality of the latest-identified matching via M_B.

1. When the latest-identified matching is M_B itself. Because M_B is a maximum matching of $B = (V_B, E_B)$ of definition 4.2, there is no augmenting path related to M_B in B by corollary 2.4, any pair of unmatched vertices of V_B^- and V_B^+ related to M_B in B, can not be the two terminals of an existing augmenting path related to M_B in $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$. Otherwise, maximality of M_B is contradicted.

2. When the latest-identified matching is bigger than M_B by one in cardinality. According to corollary 2.3 of chapter 2, there must be an augmenting path related to M_B. By case one above, this augmenting path must be either incident to $u^+ \notin \emptyset$ or $u^- \notin \emptyset$. Hence, let P_{u^-} be an existing augmenting path related to M_B and incident to $u^- \notin \emptyset$. Then, this latest-identified matching based on M_B is represented by $M_B \oplus P_{u^-}$. Meanwhile, let P_a be a path, and assume that P_a is an augmenting path related to $M_B \oplus P_{u^-}$ and its two terminals are two unmatched nodes of V_B. Related to $M_B \oplus P_{u^-}$, by case one, P_a can not alternatively only contain edges of M_B and $E_B \setminus M_B$. P_a must contain one or more edges of $P_{u^-} \setminus \{P_{u^-} \cap M_B\}$. However, there would be an augmenting path related to M_B in B, which is an contradiction. An example of P_a and P_{u^-} is shown in figure 4.2. In this figure, $P_{u^-} = \{(u^-, v_2^+), (v_2^+, v_3^-), (v_3^-, v_4^+), (v_4^+, v_5^-), (v_5^-, v_6^+), (v_6^+, v_7^-), (v_7^-, v_8^+), (v_8^+, v_9^-), (v_9^-, v_{10}^+)\}$, where those blue edges are out of M_B, and red edges belong to M_B. $P_a = \{(v_{11}^+, v_3^-), (v_3^-, v_4^+), (v_4^+, v_7^-), (v_7^-, v_8^+), (v_8^+, v_{12}^-)\}$. Also, path $\{(v_{12}^-, v_8^+), (v_8^+, v_9^-), (v_9^-, v_{10}^+)\}$ is an augmenting path related to M_B in B.

   Similarly, let P_{u^+} be an existing augmenting path related to M_B and incident to $u^+ \notin \emptyset$, for the same reason, there can not be an augmenting path related to $M_B \oplus P_{u^+}$, whose two terminals are two unmatched nodes of V_B.

3. When the latest-identified matching is bigger than M_B by two in cardinality. Based on case one above and corollary 2.3, this latest-identified matching can be obtained by either $M_B \oplus P_{u^-}$ or $M_B \oplus P_{u^+}$. For $M_B \oplus P_{u^-}$, $u^- \notin \emptyset$, u^+, according to case two above, u^+ must be unmatched by $M_B \oplus P_{u^-}$, and an existing augmenting path whose one terminal must be incident to $u^+ \notin \emptyset$. Thus, let P_{u^+} be such an augmenting path related to $M_B \oplus P_{u^-}$, and the latest-identified matching is represented by $\{M_B \oplus P_{u^-}\} \oplus P_{u^+}$. In particular, still based on case two, any edges of $P_{u^-} \setminus \{P_{u^-} \cap M_B\}$ must be still excluded by P_{u^+}. This is because there would be an augmenting path related to M_B in B.

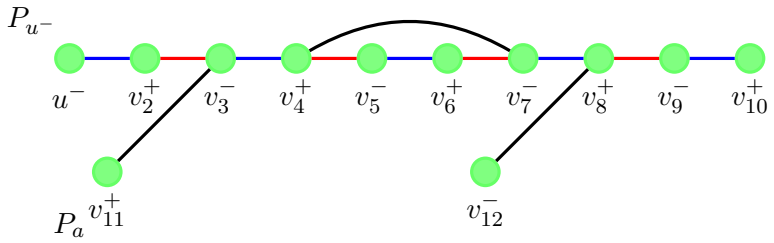

Figure 4.2:

For example, in figure 4.2, after replacing v_{11}^+ with u^+, P_{u^+} is the alternating path that involves $\{u^+, v_3^-, v_4^+, v_7^-, v_8^+, v_{12}^-\}$. Clearly, path from v_{12}^+ to v_{10}^+ is an augmenting path related to M_B in B. Therefore, P_{u^+} and P_{u^-} are also disjoint.

Next, related to $\{M_B \oplus P_{u^-}\} \oplus P_{u^+}$, assume that there is an augmenting path, whose terminals are two unmatched nodes of V_B^- and V_B^+ respectively. Because P_{u^+} and P_{u^-} are disjoint, and this augmenting path might only contain edges of $P_{u^+} \setminus \{P_{u^-} \cap M_B\}$, according to case one and two in previous, such augmenting path can not exist due to the contradiction of the maximality of M_B in B.

Similarly, for $M_B \oplus P_{u^+}$, a same result can be deduced as well.

4. By case three, related to $M_B \oplus P_{u^-} \oplus P_{u^+}$, there can not be any augmenting path related to it. Therefore, there would be no more cases to discuss according to cardinality of the latest-identified matching based on M_B.

Through discussing those four cases above, correctness of this theorem is proved.■

By theorem 4.1, corollary 4.2 is concluded to describe distribution of augmenting paths incident to $u^- \notin \emptyset$ and $u^+ \notin \emptyset$. Besides, corollary 4.3 is concluded to identify a maximum matching of $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ based on M_B.

Corollary 4.2
Given $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ of problem 4.2.2, and M_B of definition 4.2, let $P_{u^-} \neq \emptyset$ be an augmenting path related to M_B and incident to u^-. If $u^+ \notin \emptyset$ is an unmatched node related to $M_B \oplus P_{u^-}$, assume there is an augmenting path incident to u^+ and related to $M_B \oplus P_{u^-}$. Then, this augmenting path and P_{u^-} are vertex disjoint, and versa-versa.

PROOF By case three of proof of theorem 4.1, correctness of this corollary is proved.■

Corollary 4.3
Given $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ of problem 4.2.2 and M_B of definition 4.2, let P_{u^-} and P_{u^+} be two vertex-disjoint augmenting paths related to M_B incident to u^- and u^+, respectively. Then, a maximum matching of $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ is $M_B \oplus P_{u^-} \oplus P_{u^+}$.

PROOF When $P_{u^-} = \emptyset$ and $P_{u^+} = \emptyset$, by theorem 4.1, there can not be any augmenting path related to M_B, so that M_B is still a maximum matching. When $P_{u^-} \neq \emptyset$

and $P_{u^+} = \emptyset$, or $P_{u^-} = \emptyset$ and $P_{u^+} \neq \emptyset$,or $P_{u^-} = P_{u^+}$, by case two of theorem 4.1, there would be no augmenting paths related to either $M_B \oplus P_{u^-}$, or $M_B \oplus P_{u^+}$. By corollary 2.4, $M_B \oplus P_{u^-}$, or $M_B \oplus P_{u^+}$ is thus a maximum matching. When $P_{u^-} \neq \emptyset$ and $P_{u^+} \neq \emptyset$, by corollary 4.2, these two paths are disjoint, and by case three of theorem 4.1, no augmenting paths related to $M_B \oplus P_{u^+} \oplus P_u^-$ exist. In summary, by corollary 2.4, $M_B \oplus P_{u^+} \oplus P_u^-$ can thus represent a maximum matching of $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$. ∎

By corollary 4.3, given $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$, it is only needed to identify augmenting paths twice at most from $u^- \notin \emptyset$ and $u^+ \notin \emptyset$.

4.3.2 Execution

Given $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ of problem 4.2.2, and M_B of definition 4.2, algorithm 4.1 below identifies an augmenting path related to M_B and incident to u^+ if $u^+ \notin \emptyset$. Here, edges of $E_B \cup E_{B_u}$ are directed. Then, let $E_{B_u^+} \subseteq E_{B_u}$ be a set of arcs whose tails are u^+, e be any edge of $E_{B_u^+}$, and P_0 be an initially empty edge set. Also, when $P_0 \neq \emptyset$, let $P(P_0)$ be a set of edges, whose tails are heads of edges of P_0, and let $e^{'}$ be any single edge of $P(P_0)$. Besides, let P_{u^+} be an initially empty set of an augmenting path starting from u^+.

Algorithm 4.1: Identify an augmenting path incident to u^+ related to M_B.

Input : $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$, M_B, P_{u^+}.
Output: An Augmenting path starting from u^+.

Set direction of edges of M_B from nodes of V_B^- to nodes V_B^+ **and** Set direction of $\{E_B \cup E_{B_u}\} \setminus M_B$ from nodes of $V_B^+ \cup u^+$ to nodes $V_B^- \cup u^-$;
while $E_{B_u^+} \neq \emptyset$ ***and*** $e \in E_{B_u^+}$ **do**
  $P_{u^+} = \emptyset$;
  $P_{u^+}^{'} = P_{u^+} \cup e$; $E_{B_u^+}^{'} = E_{B_u^+} \setminus e$;
  if *head of e is not incident to an edge of M_B* **then**
    Set arcs of P_{u^+} as undirected edges;
    return P_{u^+}; Algorithm terminates;
    else if then
      $P_0^{'} = P_0 \cup e$ **and** Add existing arcs of E_B whose tails are head of e into $P(P_0)$;
      while $P(P_0) \neq \emptyset$ ***and*** $e^{'} \in P(P_0)$ **do**
        $P_0^{'} = P_0 \cup e^{'}$ **and** Add existing arcs of E_B whose tails are head of $e^{'}$ into $P(P_0)$;
        $P(P_0)^{'} = P(P_0) \setminus e^{'}$; $E_B^{'} = E_B \setminus e^{'}$;
        if *head of $e^{'}$ is incident to an edge out of M_B* **then**
          Identify edges of a path ending at $e^{'}$ and starting from e;
          Add identified edges into P_{u^+};
          Set arcs of P_{u^+} as undirected edges;
          return P_{u^+}; Algorithm terminates;

PROOF Initially, to traverse alternating paths from u^+, direction of edges is set in $O(|E_B \cup E_{B_u})|$ steps.

Firstly, if $E_{B_u^+} \neq \emptyset$, e is randomly selected and removed from $E_{B_u^+}$. Then, head of e is verified based on if it is not incident to an edge of M_B. If so, e can be used to augment M_B and thus added into P_{u^+}, where $P_{u^+} = \{e\}$ returned and this algorithm terminates. Otherwise, e is added into P_0, and the second **while** loop traverses all paths that start from u^+ and alternatively involve edges of $E_B \setminus M_B$ and M_B, as a way to identify an augmenting path related to M_B. When an augmenting path exists, head of currently selected edge $e^{'} \in P(P_0)$ is not incident to an edge of M_B. And this augmenting path can be identified by traversing edges based on the depth-first search [111] from $e^{'}$ to e in $O(|E_B| + |V_B|)$ time. Also, this path is added into P_{u^+}, and P_{u^+} is returned. After that, this algorithm terminates. Because the input bipartite graph is finite and each edge of $P(P_0)$ is removed from it, $P(P_0) = \emptyset$ happens at a moment and the second **while** loop terminates. Therefore, this algorithm is correct in front of the first loop.

After this, during each loop of the first **while**, P_{u^+} is emptied in line 3, because previously added edges into P_{u^+} can not construct any augmenting path. Then, following procedure is the same as before.

Finally, due to removal of each edge from $E_{B_u^+}$ in line 4, $E_{B_u^+} = \emptyset$ occurs at a moment, on which this algorithm can terminate in the end. Therefore, this algorithm is correct. ■

Corollary 4.4 (Time complexity of algorithm 4.1)
Except for obtaining inputs of algorithm 4.1, the worst-case execution time of this algorithm is $O(|E_B| + |V_B| + |E_{B_u}|)$.

PROOF For the worst-case execution time of this algorithm, due to line 12, each edge of E_B can be traversed, and added into $P(P_0)$ once at most. Meanwhile, each edge of $E_{B_u^+}$ is considered once only because of line 4. Besides, based on DFS algorithm, identifying edges of an augmenting path starting from u^+ in line 14 needs to traverse each arc of current $P(P_0)$ once at most. Also, remove direction of identified path costs $O(|E_B| + |E_{B_u}|)$ steps at most. Thus, combining the direction operation in line 1, time complexity of this algorithm is $O(|E_B| + |V_B| + |E_{B_u}|)$ except for obtaining inputs. ■

By running algorithm 4.1, it is possible that $u^- \notin \emptyset$ is another terminal of returned P_{u^+}. Nevertheless, when $u^- \notin \emptyset$ and it is still an unmatched node related to $M_B \oplus P_{u^+}$, this algorithm can be slightly modified to find an augmenting path incident to u^- and related to $M_B \oplus P_{u^+}$ in $O(|E_B|+|V_B|+|E_{B_u}|)$ time. In detail, one of inputs M_B is replaced with $M_B \oplus P_{u^+}$, which omits the process of dedicatedly identifying vertex-disjoint augmenting paths related to M_B. Besides, let $E_{B_{u^-}}$ be a set of arcs of E_{B_u} that share u^- as the common head, and P_{u^-} be an initially empty set of an augmenting path incident to u^-. Then, replace "head" with "tail", replace $E_{B_u^+}$ with $E_{B_u^-}$ and replace P_{u^+} with P_{u^-}. Also, in modified algorithm, $P(P_0)$ represents a set of edges whose heads are tails of edges of P_0. Additionally, by corollary 4.2, $P_{u^-} \cap P_{u^+} = \emptyset$.

After that, according to corollary 4.3, with P_{u^-} and P_{u^+} returned by algorithm 4.1 and the modified one, a maximum matching of $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ can be obtained, which is executed by algorithm 4.2 below:

Algorithm 4.2: Identify a Maximum Matching of $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$.

Input : $(V_B \cup \{u^-, u^+\}, E_B \cup E_{B_u})$ of problem 4.2.2, M_B of definition 4.2, P_{u^+} returned by algorithm 4.1 and P_{u^-} returned by the modified one.
Output: A maximum matching.

```
1 if P_{u^-} = ∅ and P_{u^+} = ∅ then
2     return M_B;
3     else if P_{u^-} ≠ ∅ and P_{u^+} ≠ ∅ then
4         return M_B ⊕ P_{u^-} ⊕ P_{u^+};
5     else if P_{u^-} = ∅ and P_{u^+} ≠ ∅ then
6         return M_B ⊕ P_{u^+};
7     else if P_{u^-} ≠ ∅ and P_{u^+} = ∅ then
8         return M_B ⊕ P_{u^-};
```

PROOF Based on existence of two augmenting paths incident to u^- and u^+, and according to theorem 4.1, all possible cases are listed and related maximum matching is returned by corollary 2.3. Except for running algorithm 4.1 and the modified one. The worst-case execution time of this algorithm is $O(1)$. ∎

4.3.3 Time-Complexity Analysis

Given $D = (V, E)$ with M_D of definition 4.1, after adding vertex u with edges of E_u into D, entire process of identifying a maximum matching of digraph $(V \cup \{u\}, E \cup E_u)$ or solving **Problem 1** can be shown by following algorithm 4.3, where some notations of previous algorithms are still used.

Algorithm 4.3: Identify a maximum matching of digraph $(V \cup \{u\}, E \cup E_u)$.

Input : $D = (V, E)$ with M_D of definition 4.1, $\{u, E_u\}$ of problem 4.2.2.
Output: A Maximum Matching.

```
1 Map digraph (V ∪ {u}, E ∪ E_u) with M_D into bipartite graph
    (V_B ∪ {u^-, u^+}, E_B ∪ E_{B_u}) with M_B by bijection α of definition 4.2;
2 if u^+ ∉ ∅ then
3     Run algorithm 4.1 to identify an augmenting path incident to u^- and
        related to M_B;
4 if u^- ∉ ∅ and u^+ is unmatched related to M_B ⊕ P_{u^-} then
5     Run modified algorithm 4.1 to identify an augmenting path incident to
        u^+ and related to M_B ⊕ P_{u^-};
6 Run algorithm 4.2 to identify a maximum matching of
    (V_B ∪ {u^-, u^+}, E_B ∪ E_{B_u});
7 Map derived maximum matching of line 6 by α^{-1} into an edge set of
    (V ∪ {u}, E ∪ E_u);
8 return Mapped edge set of line 7;
```

PROOF According to the correctness of algorithm 4.1 and 4.2, and the bijection of definition 4.2, this algorithm is correct. ∎

Corollary 4.5 (Time complexity of algorithm 4.3)
Except for identifying M_D of D of definition 4.1, the worst-case execution time of identifying a maximum matching of $(V \cup \{u\}, E \cup E_u)$ is $O(|E| + |V| + |E_u|)$.

PROOF The worst-case execution of this algorithm is the sum of time complexity of procedure of each line. In detail, by the bijection, time complexity of line 1 and 7 is $O(|E| + |E_u)$ for each. Then, by algorithm 4.1, time complexity of line 3, 5 respectively is $O(|E_B| + |V_B| + |E_{B_u}|)$. After line 4, time complexity of line 5 by algorithm 4.2 is $O(1)$. Furthermore, because $|E_B| = |E|$ by definition 4.2 and $|E_u| = |E_{B_u}|$. Except for identifying M_D of D of definition 4.1, the worst-case execution time of identifying a maximum matching of $(V \cup \{u\}, E \cup E_u)$ is therefore $O(|E| + |V| + |E_u|)$. ∎

Clearly, the worst-case execution time of solving the problem 4.2.2 is $O(|E| + |V| + |E_u|)$, which excludes obtaining M_D.

4.3.4 Comparison

Compared with related works about structural-controllability recovery in section 3.2.1 of chapter 3, our assumptions on the input network are contained by them, while the type of the input network, availability of tree decomposition constrained by related works [9] are not assumed in this chapter. Also, except for acquiring M_D of $D = (V, E)$ of definition 4.1, as we can see, algorithm 4.3 runs in linear time, which is a lower time complexity than that of those related works.

Simultaneously, with the assumption that $|E_u| \ll |E|$ of section 4.2.2, once concerning acquiring the initial structural controllability with a minimum set of inputs, by finding a maximum matching, the worst-case execution time is just $O(\sqrt{|V|} \cdot |E|)$. Furthermore, referring the result of [17], if D is a sparse ER random digraph, the average-case time complexity of solving the **Problem 1** is $O(|E| \cdot log(|V|))$.

Additionally, viewing the entire execution process, because the maximum matching of $(V_B \cup \{u^-, u^+, E_B \cup E_{B_u}\})$ is obtained by augmenting M_B of B through identifying augmenting path related to M_B, according to theorem 4.1 and corollary 4.2. Unmatched nodes related to M_B in B can thus still be unmatched related to the identified one in $(V_B \cup \{u^+, u^-\}, E_B \cup E_{B_u})$ with maximum number. Therefore, by definition 4.2 and the minimum input theorem, the maximum number of previous inputs would be reused to recover structural control into D after adding a single node.

4.4 Summary

Focusing on structural-controllability recovery after very limited modification on system component or network vertex with CT-LTI dynamics, this chapter solves the problem of recovering structural controllability of an initially minimum-input structurally controllable CT-LTI network after adding a single vertex. According to maximum-matching based method of section 2.4.1 of chapter 2, we identify a maximum matching of resulting graph without recomputation, and the worst-time execution time is linear except for computing a maximum matching for original

minimum set of inputs. Additionally, on the other hand of very-limited modification, chapter 5 would concentrate on the recovery after removing a precomputed system component by identifying a maximum matching without recomputation.

Chapter 5

Structural-Control Recovery for Resilient Control Systems

5.1 Overview

Following chapter 4, this chapter still concentrates on structural-controllability recovery after very-limited modification on CT-LTI system components. Through section 1.2.4 of chapter 1, and the simulation of [96], we already know that once information communication among sensors, controllers and actuators of a control system is attacked or failed, this control system could lose control into its physical system immediately. Nevertheless, because physical system is another most necessary part of a control system, a control system can still lose its control due to sabotage of the physical system, even if other parts are still working well. Clearly, to maintain a resilient control system against malicious attack or random failure on a single component of a physical system, it is also desirable to recover the controllability of the residual physical system with high efficiency.

Therefore, to enhance resilience of a control system, this chapter solves research question 2. Specifically, with assumptions of section 1.3.2 of chapter 1, this chapter efficiently recovers structural control into a minimum-input structurally controllable physical system, after removing an already known system component. Referring to related works about robustness of network structural controllability of section 3.3.1 of chapter 3, since some removed network vertex can be confirmed by its importance, such as its degree, betweenness and so on, this removed system component can be also confirmed or identified via the similar way, while this confirmation is out of discussion of this chapter. Besides, we also assume that structural controllability of this given physical system is acquired by the maximum-matching based method of section 2.4.1 of chapter 2.

For the solution, according to Lin's graphic interpretation of the state matrix, we define a digraph as our input network to represent the state matrix of the given control system, which is like $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4 of chapter 2. Also, we assume that this digraph contains a known maximum matching. Then, by corollary 2.5 of chapter 2, this problem is thus transferred into identifying a maximum matching of the given digraph after removing a known vertex without recomputation, which is eventually solved in this chapter. Based on our inference, this graph-theoretical problem is further solved by identifying a maximum matching of the input network, which must contain the minimum number of edges incident to the removed vertex. As a result, a maximum matching of the residual network can be identified in linear time except for computing the known maximum matching of the input network.

For the contribution of this chapter, given any initially minimum-input struc-

turally controllable CT-LTI physical system, structural-control recovery to against a single component removal, can be done in linear time in the worst case. Besides, original inputs can be also reused with maximum number. Again, this result also reflects that structural-controllability recovery should concern the amount of change on the initial network.

This chapter is structured as follows: section 5.2 defines an input network and formulates research question; section 5.3 shows the solution and section 5.4 concludes this chapter.

5.2 Problem Formulation

Given an initially minimum-input structurally controllable physical system, which is described by following equation:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{5.1}$$

where, $\mathbf{A} \in \mathbb{R}^{n\times n}$, and the number of columns of matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ is minimum. Then, let $\mathbf{A}^{'} \in \mathbb{R}^{(n-1)\times(n-1)}$ be a state matrix of the given system after removing a known system component. Particularly, $\mathbf{A}^{'}$ is obtained by removing both ith row and ith column from matrix $\mathbf{A}$ above, where $1 \leq i \leq n$. Besides let $\mathbf{B}^{'}$ be an input matrix. Above all, our research question is formally specified:

Research Question: Given $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{A}^{'}$. Then, efficiently identify an input matrix $\mathbf{B}^{'}$ with the minimum number of columns, so that the dynamical system described by $\dot{x}(t) = \mathbf{A}^{'}x(t) + \mathbf{B}^{'}u(t)$ is structurally controllable.

Then, an input network is defined to model research question:

Definition 5.1 (Input Network of chapter 5)
Let $D = (V, E)$ be a large, finite digraph, and D excludes self loops, parallel arcs and isolated nodes. Also, V represents the vertex set, $V = \{v_i | 1 \leq i \leq n\}(n > 2)$, and E represent the edge set, $E = \{\overrightarrow{\langle v_i, v_j \rangle} | v_i, v_j \in V\}$. Besides, let M_D be a fixed and arbitrary maximum matching of D, identified by the algorithm of [60].

With $D = (V, E)$ and known maximum matching M_D of definition 5.1, assume that D is mapped by original state matrix $\mathbf{A}$ of equation (5.1) according to definition 2.4 of chapter 2, where any arc of E corresponds to only one non-zero entry of $\mathbf{A}$. Now, research question of this chapter is modelled by following graph-theoretical problem:

Problem 1: Given $D = (V, E)$ with M_D, let $u \in V$ be a vertex that is pre-identified and to be removed from D, and let E_u be a set of all arcs incident to u, where $|E_u| \ll |E|$. Then, efficiently identify a maximum matching of digraph $D \setminus \{u\}$ without recomputation.

Problem 5.2 is solved in following section as a way to address the research question from graph-theoretical aspect.

5.3 Solution

5.3.1 Maximum-matching Identification within $D \setminus \{u\}$

Without recomputing a maximum matching of digraph $D \setminus \{u\}$, if u is neither head nor tail of an arc of M_D, it is obviously enough to say that M_D is a maximum matching of $D \setminus \{u\}$. Otherwise, we also deduce lemma 5.1 and lemma 5.3 to identify a maximum matching of $D \setminus \{u\}$ when u is incident to an edge of M_D.

In the following paper, let MM be a maximum matching of D, and MM contains the minimum number of arcs incident to u. With MM, let l be the minimum number of edges incident to u. Clearly, there is $l \in \{0, 1, 2\}$. Besides, let $\overrightarrow{\langle v_g, u\rangle}$ and $\overrightarrow{\langle u, v_f\rangle}$ be two such arcs of MM. When $\overrightarrow{\langle v_g, u\rangle} \in MM$ and $\overrightarrow{\langle u, v_f\rangle} \in MM$ together, let $M_{(v_g,v_f)}$ be a matching and $M_{(v_g,v_f)} \cap MM = \emptyset$, where v_g is a tail of an arc of $M_{(v_g,v_f)}$, and v_f is a head an arc of $M_{(v_g,v_f)}$, respectively. Also, each arc of $M_{(v_g,v_f)}$ shares a common head with an arc of MM, and also shares a common tail with another different arc of MM.

Lemma 5.1
Given $D = (V, E)$ of definition 5.1, MM with $l \in \{0, 1, 2\}$ and $M_{(v_g,v_f)} = \emptyset$. Then, the maximum matching of $D \setminus \{u\}$ is the matching that is MM excluding arcs incident to u.

PROOF Correctness of this lemma is proved when $l = 0, l = 1, l = 2$ respectively:

1. If $l = 0$, there is no arc of MM incident to u. Then, removing u from D does not influence MM. Thus, MM still exists in $D \setminus \{u\}$ as a maximum matching.

2. If $l = 1$, it is either $\overrightarrow{\langle v_g, u\rangle} \in MM$ or $\overrightarrow{\langle u, v_f\rangle} \in MM$. After removing u, a matching $MM \setminus \overrightarrow{\langle v_g, u\rangle}$, or $MM \setminus \overrightarrow{\langle u, v_f\rangle}$ is obtained directly. Assume that this matching is not a maximum matching of $D\setminus\{u\}$, we should be able to identify a matching with the cardinality of $|MM|$ through matching $MM \setminus \overrightarrow{\langle v_g, u\rangle}$, or $MM \setminus \overrightarrow{\langle u, v_f\rangle}$. Actually, it needs to identify a matching with bigger cardinality than a subset of MM, and this matching must be adjacent to edges of MM at both heads and tails. Otherwise, if this proposed matching excludes any single edge that is not adjacent to edges of remaining MM, cardinality of MM would be contradicted. Thus, following two cases are discussed when $l = 1$ according to if the purposed matching is incident to v_f or v_g:

   On the one hand, in $D \setminus \{u\}$, there could be a matching that excludes any edge of MM and only involves edges that are not incident to v_f or v_g, where each edge of a subset of MM shares a common head and tail with two arcs of this matching. For this situation, maximality of MM is thus contradicted. For example, in figure 5.1, a blue directed path starting from u to v_6 is contained by MM on assumption, where v_f is v_2, and $\{\overrightarrow{\langle v_3, v_7\rangle}, \overrightarrow{\langle v_6, v_4\rangle}\}$ is the proposed matching. However, the directed path from u to v_7 and the directed cycle containing v_4, v_5 and v_6 is a bigger matching than the blue path in cardinality.

   On the other hand, in $D \setminus \{u\}$, there could be a matching excluding any edge of MM and it contains an arc whose head is v_f or whose tail is v_g, where each edge of a subset of MM shares a common head and tail with two arcs

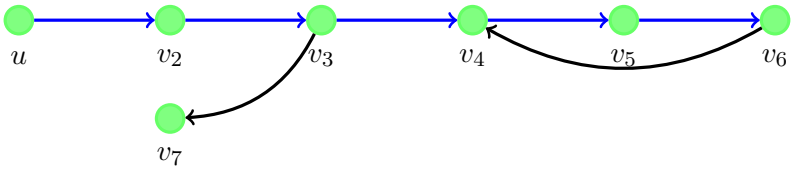

Figure 5.1:

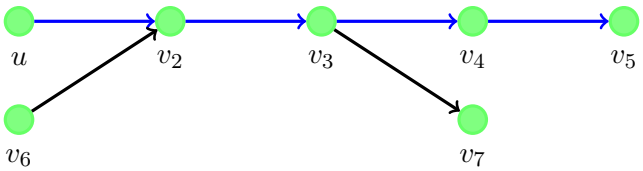

Figure 5.2:

of this matching. In this case, $l = 1$ is contradicted. As a simple example of this case, in figure 5.2, that blue-colour directed path from u to v_5 is a subset of MM and disjoint with other paths and cycles of MM, where v_f is v_2, and $\{\overrightarrow{\langle v_6, v_2\rangle}, \overrightarrow{\langle v_3, v_7\rangle}\}$ is a proposed matching. However, because arc $\overrightarrow{\langle u, v_2\rangle}$ can be replaced with $\overrightarrow{\langle v_6, v_2\rangle}$, which is involved in the matching $\{\overrightarrow{\langle v_6, v_2\rangle}, \overrightarrow{\langle v_2, v_3\rangle}, \overrightarrow{\langle v_3, v_7\rangle}\}$.

Above all, since there can not be such a matching bigger than either $MM \setminus \overrightarrow{\langle v_g, u\rangle}$, or $MM \setminus \overrightarrow{\langle u, v_f\rangle}$ in cardinality, $MM \setminus \overrightarrow{\langle v_g, u\rangle}$, or $MM \setminus \overrightarrow{\langle u, v_f\rangle}$ is thus a maximum matching of $D \setminus \{u\}$.

3. If $l = 2$ and $M_{(v_g, v_f)} = \emptyset$, there are $\overrightarrow{\langle v_g, u\rangle} \in MM$ and $\overrightarrow{\langle u, v_f\rangle} \in MM$. Then, a matching $MM \setminus \{\overrightarrow{\langle v_g, u\rangle}, \overrightarrow{\langle u, v_f\rangle}\}$ is obtained directly. According to those two discussed situations of case 2 above, and also because of $M_{(v_g, v_f)} = \emptyset$, it can be concluded that $MM \setminus \{\overrightarrow{\langle v_g, u\rangle}, \overrightarrow{\langle u, v_f\rangle}\}$ is a maximum matching of $D \setminus \{u\}$.

Above all, because each case indicates that the matching derived by MM excluding arcs incident to u is a maximum matching of $D \setminus \{u\}$, this lemma is thus correct. ∎

According to lemma 5.1, corollary 5.2 is concluded to clarify the identification of a matching via a subset of MM:

Corollary 5.2
Given $D = (V, E)$ of definition 5.1, MM and u. Then, in $D \setminus u$, any existing subset of MM can not be used to identify a matching that is bigger than it, so that it is impossible to obtain a bigger matching in this way than remaining MM in cardinality.

PROOF Based on the contradictions of proof of lemma 5.1, which are clearly indicated by case two, correctness of this corollary is thus proved. ∎

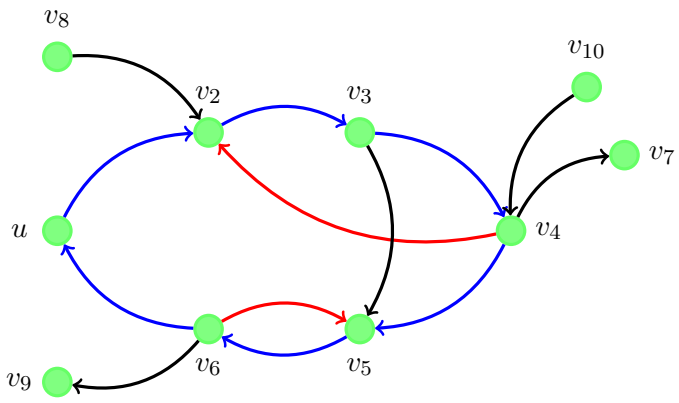

Figure 5.3:

Lemma 5.3
Given $D = (V, E)$ of definition 5.1, MM with $l = 2$, and $M_{(v_g,v_f)} \neq \emptyset$, let M_{sub} be a subset of MM, and assume that each arc of M_{sub} shares a common head and tail with two different arcs of $M_{(v_g,v_f)}$ except for v_g and v_f, where $|M_{sub}| = |M_{(v_g,v_f)}| - 1$. Then, the maximum matching of $D \setminus \{u\}$ is $MM \setminus \{\{\overrightarrow{\langle u, v_f \rangle}, \overrightarrow{\langle v_g, u \rangle}\} \cup M_{sub}\} \cup M_{(v_g,v_f)}$.

PROOF If $l = 2$, u is either contained by a directed path or a directed cycle of MM. With $M_{(v_g,v_f)} \neq \emptyset$ and M_{sub}, a matching $MM \setminus \{\{\overrightarrow{\langle u, v_f \rangle}, \overrightarrow{\langle v_g, u \rangle}\} \cup M_{sub}\} \cup M_{(v_g,v_f)}$ is directly obtained in $D \setminus \{u\}$. Assume that this matching is not a maximum matching. Nevertheless, since the remaining subset of MM in this matching is equal to the subset of MM of corollary 5.2. And any subset of MM can not be used to identify bigger matching that contains edges incident to nodes out of MM in cardinality than remaining MM that excludes u. It is thus possible to identify a matching only based on subset of $M_{(v_g,v_f)}$, where an edge of $M_{(v_g,v_f)}$ should share common head and tail with two different edges of this proposed matching. In this case, $l = 2$ contradicted. For example, in figure 5.3, the blue directed cycle is a subset of MM and disjoint with other ones, where $v_g = v_6, v_f = v_2$ and $M_{(v_g,v_f)} = \{\overrightarrow{\langle v_6, v_5 \rangle}, \overrightarrow{\langle v_4, v_2 \rangle}\}$ and $M_{sub} = \{\overrightarrow{\langle v_4, v_5 \rangle}\}$. Then, for $\overrightarrow{\langle v_8, v_2 \rangle} \neq \emptyset$ and $\overrightarrow{\langle v_4, v_7 \rangle} \neq \emptyset$, u as a tail could be excluded by the path from v_8 to u, which means $l = 1$ and is a contradiction with $l = 2$. Besides, it is also possible to identify a matching based on both a subset of MM and $M_{(v_g,v_f)}$. Again, by the same contradiction mentioned before, this case is also impossible. For example, still in figure 5.3, based on the matching $\{\overrightarrow{\langle v_6, v_5 \rangle}, \overrightarrow{\langle v_3, v_4 \rangle}\}$, such a proposed matching is $\{\overrightarrow{\langle v_6, v_9 \rangle}, \overrightarrow{\langle v_3, v_5 \rangle}, \overrightarrow{\langle v_{10}, v_4 \rangle}\}$. Clearly, in this example, $l \neq 2$.

Above all, with conditions of this lemma, there can not be a matching identified through a subset of $M_{(v_g,v_f)}$ to derive a matching with bigger cardinality than matching $MM \setminus \{\{\overrightarrow{\langle u, v_f \rangle}, \overrightarrow{\langle v_g, u \rangle}\} \cup M_{sub}\} \cup M_{(v_g,v_f)}$. Thus, $MM \setminus \{\{\overrightarrow{\langle u, v_f \rangle}, \overrightarrow{\langle v_g, u \rangle}\} \cup M_{sub}\} \cup M_{(v_g,v_f)}$ is a maximum matching of $D \setminus \{u\}$. ∎

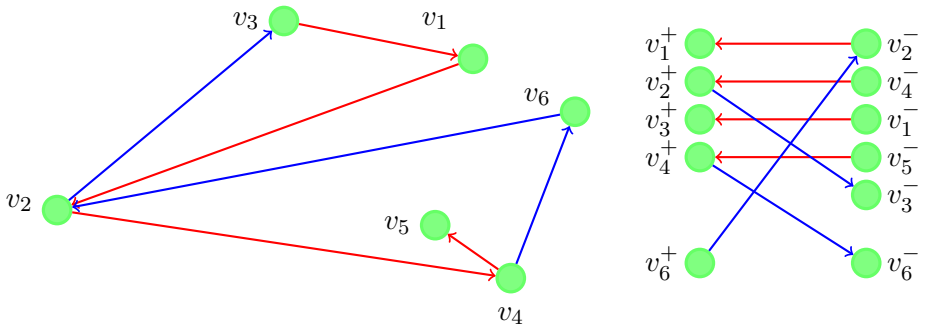

Figure 5.4:

A digraph contains a maximum matching that is a red path, which is mapped into a directed bipartite graph by definition 5.2 with a maximum matching involving all red arcs.

5.3.2 Identification of MM

Given $D = (V, E)$ with M_D of definition 5.1 and $u \in V$, by lemma 5.1 and 5.3, identifying MM is the key to confirm a maximum matching of $D \setminus \{u\}$. To identify MM of D, we use a directed bipartite graph defined below:

Definition 5.2
Given $D = (V, E)$ with M_D, let $B = (V_B, E_B)$ be a directed bipartite graph, V_B^+ and V_B^- be two disjoint and independent sets of V_B, where $|E_B| = |E|$, $|V_B| \leq 2|V|$, and $V_B = \{\{v_i^-, v_j^+\} | v_i^- \in V_B^-, v_j^+ \in V_B^+\}$. Besides, let M_B be a maximum matching of B. Then, let $\alpha : E \setminus M_D \to E_B \setminus M_B$, and $\beta : M_D \to M_B$ be two different bijections. For any $\overrightarrow{\langle v_i, v_j \rangle} \in E \setminus M_D$, $\alpha : \overrightarrow{\langle v_i, v_j \rangle} \to \overrightarrow{\langle v_i^+, v_j^- \rangle}$, where $\overrightarrow{\langle v_i^+, v_j^- \rangle} \in E_B \setminus M_B$; for any $\overrightarrow{\langle v_p, v_q \rangle} \in M_D$, $\beta : \overrightarrow{\langle v_p, v_q \rangle} \to \overrightarrow{\langle v_q^-, v_p^+ \rangle}$, where $\overrightarrow{\langle v_q^-, v_p^+ \rangle} \in M_B$.

Any edge like $\overrightarrow{\langle v_i^+, v_i^- \rangle}$ can not exist in B, otherwise D includes the selfloop, and an example of bijections of definition 5.2 is shown by figure 5.4:

In $B = (V_B, E_B)$, let $u^- \in V_B^-$ and $u^+ \in V_B^+$ be mapped by $u \in V$ through bijections α or β. Therefore, identifying MM of D can be solved by identifying a maximum matching of B that contains the minimum number of arcs incident to u^- and u^+. To do this, theorem 5.4 and 5.5 are concluded, where $\overrightarrow{\langle u^-, v_p^+ \rangle}$ and $\overrightarrow{\langle v_q^-, u^+ \rangle}$ are defined as two arcs of M_B by definition 5.2 in the following paper. Besides, let $M_B^{'}$ be any different maximum matching from M_B and $|M_B| = |M_B^{'}|$.

Theorem 5.4
Given $B = (V_B, E_B)$ with M_B of definition 5.2, and $\{u^-, u^+\} \subseteq V_B$. Then, $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$ and $\overrightarrow{\langle u^-, v_p^+ \rangle} \notin M_B^{'}$, if and only if there is a directed augmenting path related to $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$, which starts from v_p^+.

PROOF $\Rightarrow$: If $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$ and $\overrightarrow{\langle u^-, v_p^+ \rangle} \notin M_B^{'}$. Given $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$, by corollary 2.3, there must be an directed augmenting path related to it, otherwise, $M_B^{'}$ can not exist. Since v_p^+ now is an unmatched node related to matching $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$,

and any other node of V_B^+ that are not incident to arcs of M_B can not be a starting node of such an augmenting path, otherwise, maximality of M_B is contradicted. Therefore, v_p^+ must be the only one starting node of such an augmenting path.

$\Leftarrow$: If there is a directed augmenting path related to $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$, which starts from v_p^+. By the symmetric difference between this augmenting path and $M_B \setminus \{\overrightarrow{\langle u^-, v_p^+ \rangle}\}$, a maximum matching excluding an edge incident to u^- can be obtained, which is thus noted by $M_B^{'}$. ■

By theorem 5.4, it can be also concluded that $\overrightarrow{\langle v_q^-, u^+ \rangle} \in M_B$ and $\overrightarrow{\langle v_q^-, u^+ \rangle} \notin M_B^{'}$ if and only if there is a directed augmenting path related to $M_B \setminus \overrightarrow{\langle v_q^-, u^+ \rangle}$ and ending at v_q^-.

Theorem 5.5
Given $B = (V_B, E_B)$ with M_B of definition 5.2, $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$ and $\overrightarrow{\langle v_q^-, u^+ \rangle} \in M_B$. Then, in $B \setminus \{u^-, u^+\}$, related to $M_B \setminus \{\overrightarrow{\langle u^-, v_p^+ \rangle}, \overrightarrow{\langle v_q^-, u^+ \rangle}\}$, an available augmenting path starting from v_p^+ and another different augmenting path ending at v_q^- must be vertex disjoint.

PROOF Let $M_B^{'}$ be a maximum matching of B, and let $\{u^-, u^+\}$ be not incident to any arc of $M_B^{'}$. Because $M_B^{'} \oplus M_B$ only includes vertex-disjoint directed paths or cycles with even length, which alternatively involve arcs of $M_B^{'}$ and M_B. Hence, within $M_B^{'} \oplus M_B$, an available directed path starting from u^- and an available directed path ending at u^+ can not share any vertex in B. Then, after removing $\{u^-, u^+\}$, as a result, two vertex-disjoint augmenting paths related to $M_B \setminus \{\overrightarrow{\langle u^-, v_p^+ \rangle}, \overrightarrow{\langle v_q^-, u^+ \rangle}\}$ are produced, one of which either starts from v_p^+ or ends at v_q^-. For example, in figure 5.4, let $v_1^+ = u^+$ and $v_4^- = u^-$, there are two vertex disjoint paths, $\{\overrightarrow{\langle v_6^+, v_2^- \rangle}, \overrightarrow{\langle v_2^-, v_1^+ \rangle}\}$ and $\{\overrightarrow{\langle v_4^-, v_2^+ \rangle}, \overrightarrow{\langle v_2^+, v_3^- \rangle}\}$. After removing $\{v_1^+, v_4^-\}$, $\overrightarrow{\langle v_6^+, v_2^- \rangle}$ and $\overrightarrow{\langle v_2^+, v_3^- \rangle}$ are two vertex-disjoint augmenting paths related to matching of remaining red arcs. ■

5.3.3 Execution

To begin with, a maximum matching of $B = (V_B, E_B)$ of definition 5.2 that contains the minimum number of arcs incident to $\{u^-, u^+\}$ is noted by $M_B^{'}$, and it would be identified by following algorithm 5.1. Then, by algorithm 5.2, $M_B^{'}$ is further used to identify the maximum matching MM of the digraph $D = (V, E)$ of definition 5.1. After that, problem 5.2 of section 5.2 is eventually solved by algorithm 5.3 and 5.4.

5.3.3.1 Derive $M_B^{'}$

Algorithm 5.1 below would identify a directed augmenting path starting from v_p^+ and related to $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$ firstly, when $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$. And a maximum matching of B contains the minimum number of edges incident to u^- can be further obtained. Here, let $E_{B_{v_p^+}}$ be a set of arcs whose tails are v_p^+, e be any edge of $E_{B_{v_p^+}}$, and P_0 be an initially empty edge set. Also, with $P_0 \neq \emptyset$, let $P(P_0)$ be a set of edges,

whose tails are heads of edges of P_0, and let $e^{'}$ be any single edge of $P(P_0)$. Besides, $P_{v_p^+}$ represents an initially empty set of an augmenting path starting from v_p^+.

Algorithm 5.1: Find a maximum matching with minimum number of edges incident to u^-.

Input : $B = (V_B, E_B)$ and M_B of definition 5.2, $\{u^-, u^+\}$ mapped by u of problem 5.2.

Output: A maximum matching excluding an arc incident to u^-.

1 **while** $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$, $E_{B_{v_p^+}} \neq \emptyset$ ***and*** $e \in E_{B_{v_p^+}}$ **do**
2 $P_{v_p^+} = \emptyset$;
3 $P^{'}_{v_p^+} = P_{v_p^+} \cup e$; $E^{'}_{B_{v_p^+}} = E^{'}_{B_{v_p^+}} \setminus e$;
4 **if** *head of e is not tail of the arc of M_B* **then**
5 **return** $M^{'}_B = M_B \oplus \{\overrightarrow{\langle u^-, v_p^+ \rangle} \cup P_{v_p^+}\}$; Algorithm terminates;
6 **else if then**
7 $P^{'}_0 = P_0 \cup e$ **and** Add existing arcs of E_B whose tails are head of e into $P(P_0)$;
8 **while** $P(P_0) \neq \emptyset$ ***and*** $e^{'} \in P(P_0)$ **do**
9 $P^{'}_0 = P_0 \cup e^{'}$ **and** Add existing arcs of E_B whose tails are head of $e^{'}$ into $P(P_0)$;
10 $P(P_0)^{'} = P(P_0) \setminus e^{'}$; $E^{'}_B = E_B \setminus e^{'}$;
11 **if** *head of $e^{'}$ is not incident to the edge of M_B* **then**
12 Find a path ending at $e^{'}$ and starting from e within P_0;
13 Add found path into $P_{v_p^+}$;
14 **return** $M^{'}_B = M_B \oplus \{\overrightarrow{\langle u^-, v_p^+ \rangle} \cup P_{v_p^+}\}$; Algorithm terminates;
15 **return** $M^{'}_B = M_B$;

PROOF In the beginning, with M_B and u^-, traversing edges of M_B can confirm if $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$ in $O(|E_B|)$ steps at most. If so, v_p^+ is used to identify an augmenting path starting from it and related to $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$, and the matching obtained by this augmenting path is a maximum matching by theorem 5.4.

In front of the first iteration of the **while** loop of line 1, an arc $e \in E_{B_{v_p^+}}$ is randomly selected to identify an augmenting path, on which $P_{v_p^+} = \{e\}$. If head of e is not tail of an edge of M_B, e is thus an augmenting path related to $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$, and maximum matching $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle} \oplus e$ is returned, and this algorithm is terminated. Otherwise, second **while** loop is triggered to keep searching. During this procedure, with P_0 and $P(P_0)$, all arcs of E_B that can be approached by v_p^+ can be traversed once only, because any firstly traversed edge is removed from E_B in line 10. Therefore, second loop can terminate. Until the existence of $e^{'} \in P(P_0)$, whose head is not incident to an edge of M_B, an augmenting path starting from v_p^+ and related to $M_B \setminus \overrightarrow{\langle u^-, v_p^+ \rangle}$ exists. Then, in line 12, identifying a path between e

and $e^{'}$ in $O(|E_B|)$ time can obtain such an augmenting path. In detail, it just requires a depth-first search that starts from $e^{'}$ until e is visited. And a maximum matching not incident to u^- is therefore obtained, by which, this algorithm is terminated. Particularly, such path identification needs to traverse edges of this path twice at most. Above all, this algorithm is correct before the first iteration.

If this algorithm is not terminated in front of the first iteration, an arc of remaining $E_{B_{v_p^+}}$ is chosen to keep searching as before. Because of the line 2, any $P_{v_p^+}$ operated before this iteration is emptied now, the augmenting-path search is completely same as before until an augmenting path is found or this algorithm terminates in the end. Moreover, if there is no such an augmenting path starting from v_p^+ after the last iteration of the first **while** loop, this algorithm returns M_B, which means that each maximum matching of B must contain $\overrightarrow{\langle u^-, v_p^+ \rangle}$.

Further, because each chosen edge of $E_{B_{v_p^+}}$ and traversed edge of E_B from v_p^+ are all removed directly, this algorithm must terminate when $E_{B_{v_p^+}} = \emptyset$. Therefore, this algorithm is also correct during each iteration. ■

Corollary 5.6 (Time complexity of algorithm 5.1)
Except for deriving $B = (V_B, E_B)$ and M_B, the worst-case execution time of running algorithm 5.1 is $O(|E_B|)$.

PROOF For the worst-case execution time, along with paths starting from u^-, because each edge of E_B is traversed twice at most. Once is cost by obtaining P_0, another is cost by traversing edges of P_0. Running the second **while** loop thus costs $O(|E_B|)$ time. Combined with other operations mentioned above, time complexity of this algorithm is $O(|E_B|)$ except for deriving M_B and $\{u^-, u^+\}$. ■

Later, by slightly modifying algorithm 5.1, when $\overrightarrow{\langle v_q^-, u^+ \rangle} \in M_B$, a maximum matching of B that excludes any arc incident to u^+ can be identified with the worst-case execution time $O(|E_B|)$. Specifically, let $E_{B_{v_q^-}}$ be a set of arcs of E_B whose heads are v_q^-, and $P_{v_q^-}$ be an initially empty set of an augmenting path ending at v_q^-. Then, given algorithm 5.1, replace "head" with "tail", replace $E_{B_{v_p^+}}$ with $E_{B_{v_q^-}}$, and replace $P_{v_p^+}$ with $P_{v_q^-}$. Also, in modified algorithm, $P(P_0)$ represents a set of edges whose heads are tails of edges of P_0. And the return of this modified algorithm is either $M_B \oplus \{\overrightarrow{\langle v_q^-, u^+ \rangle} \cup P_{v_q^-}\}$, or M_B. Additionally, if $\overrightarrow{\langle u^-, v_p^+ \rangle} \in M_B$ and $\overrightarrow{\langle v_q^-, u^+ \rangle} \in M_B$ together, by theorem 5.5, related to $M_B \setminus \{\overrightarrow{\langle u^-, v_p^+ \rangle}, \overrightarrow{\langle v_q^-, u^+ \rangle}\}$, for any identified two augmenting paths, each of which either starts from v_p^+ or ends at v_q^-, must be vertex disjoint.

In summary, $M_B^{'}$ can be thus represented by one of following equations:

1. $M_B^{'} = M_B$
2. $M_B^{'} = M_B \oplus \{\overrightarrow{\langle u^-, v_p^+ \rangle} \cup P_{v_p^+}\}$
3. $M_B^{'} = M_B \oplus \{\overrightarrow{\langle v_q^-, u^+ \rangle} \cup P_{v_q^-}\}$
4. $M_B^{'} = M_B \oplus \{\overrightarrow{\langle v_q^-, u^+ \rangle} \cup P_{v_q^-}, \overrightarrow{\langle u^-, v_p^+ \rangle} \cup P_{v_p^+}\}$

5.3.3.2 Derive MM

Above all, algorithm 5.2 returns the matching MM of $D = (V, E)$ of definition 5.1, which contains the minimum number of arcs incident to u. Here, let $e = \overrightarrow{\langle v_i^*, v_j^{-(*)} \rangle}$ be any arc of $M_B^{'}$, where $*$ represents either $+$ or $-$.

Algorithm 5.2: Derive a maximum matching MM of D.

Input : $D = (V, E)$ with M_D of definition 5.1, u.
Output: maximum matching MM of D.

```
1  MM = ∅;
2  if there is no arcs of M_D incident to u then
3      MM' = MM ∪ M_D;
4      else if then
5          Derive B = (V_B, E_B) with M_B by definition 5.2;
6          Identify M'_B by using algorithm 5.1;
7          if M'_B ≠ M_B then
8              while M'_B ≠ ∅ and e ∈ M'_B do
9                  Remove e from M'_B;
10                 if * is − then
11                     MM' = MM ∪ ⟨v_j, v_i⟩⃗;
12                 else if then
13                     MM' = MM ∪ ⟨v_i, v_j⟩⃗;
14             else if then
15                 MM' = MM ∪ M_D;
16 return MM;
```

PROOF Initially, MM is set as an empty set. Then, checking if u is incident to arcs of M_D or not, which can be done by checking tail and head of each arc of M_D in $O(|E|)$ time for $D = (V, E)$. If not, MM is directly set as M_D in line 3 and returned in line 16. Otherwise, bipartite graph B with M_B are derived in line 5 and 6, in order to identify $M_B^{'}$ by using algorithm 5.1. If $M_B^{'} \neq M_B$, it is used to derive MM in the **while** loop. In detail, for each arc of $M_B^{'}$, its direction is checked in order to confirm the relative arc of D in either procedure of line 10 or 12, which is based on the bijections of definition 5.2. Due to each arc of $M_B^{'}$ is chosen and removed, $M_B^{'} = \emptyset$ terminates this **while** loop. If $M_B^{'} = M_B$, it means that M_D is the maximum matching that contains the minimum number of arcs incident to u. And MM is set as M_D. According to those possibilities, MM is finally returned. ∎

Corollary 5.7 (Time complexity of algorithm 5.2)
Except for obtaining $D = (V, E)$ and M_D, the worst-case execution time of running algorithm 5.2 is $O(|E|)$.

PROOF For the time complexity, except for obtaining u, M_D, deriving B and M_B by definition 5.2 costs $O(|E|)$ time. Also, by algorithm 5.1, identifying $M_B^{'}$ costs $O(|E_B|)$ time. Besides, without repeatedly checking arc of $M_B^{'}$, time complexity of

while loop is $O(|E|)$. Because $|E| = |E_B|$ by definition 5.2, time complexity of this algorithm is $O(|E|)$. ■

5.3.3.3 Identify $M_{(v_g,v_f)}$

After that, by lemma 5.3, with MM and $l = 2$, to obtain a maximum matching of $D \setminus \{u\}$, it is still essential to confirm if $M_{(v_g,v_f)}$ exists or not, when $\overrightarrow{\langle u, v_f \rangle} \in MM$ and $\overrightarrow{\langle v_g, u \rangle} \in MM$. This confirmation is done by algorithm 5.3 below. In this algorithm, by those bijections of definition 5.2, D and MM are mapped into a directed bipartite graph with a maximum matching, noted by $B^{'}$ and MM_B, and there must be $\overrightarrow{\langle v_f^-, u^+ \rangle} \in MM_B$ and $\overrightarrow{\langle u^-, v_g^+ \rangle} \in MM_B$. Here, let M_{sub} be an initially empty subset of MM of D, and let $M_{(v_g,v_f)}$ be an initially empty set. Besides, let $T(v_g^+)$ be a set of arcs whose tails are v_g^+, and let each arc of $T(v_g^+)$ be e. Additionally, P_0, $e^{'} \in P(P_0)$ of original algorithm 5.1 are still used here.

Algorithm 5.3: Derive a matching $M_{(v_g,v_f)}$.

Input : $D = (V, E)$ of definition 5.1, MM returned by algorithm 5.2, v_f, v_g, M_{sub}, $M_{(v_g,v_f)}$.
Output: Matching $M_{(v_g,v_f)}$.

1 Map $D = (V, E)$ and MM into $B^{'}$ with MM_B by bijections of definition 5.2;
2 **while** $T(v_g^+) \neq \emptyset$ ***and*** $e \in T(v_g^+)$ **do**
3 $P_0 = \emptyset$;
4 $T(v_g^+)^{'} = T(v_g^+) \setminus e$;
5 **if** *Head of* e *is* v_f^- **then**
6 **return** $M_{(v_g,v_f)} = \overrightarrow{\langle v_g, v_f \rangle}$, $M_{sub} = \emptyset$; Algorithm terminates;
7 **else if then**
8 $P_0^{'} = P_0 \cup e$ **and** Add existing arcs of $B^{'}$ whose tails are head of e into $P(P_0)$;
9 **while** $P(P_0) \neq \emptyset$ ***and*** $e^{'} \in P(P_0)$ **do**
10 $P(P_0)^{'} = P(P_0) \setminus e^{'}$; Remove $e^{'}$ from $B^{'}$;
11 $P_0^{'} = P_0 \cup e^{'}$ **and** Add existing arcs of $B^{'}$ whose tails are head of $e^{'}$ into $P(P_0)$;
12 **if** *Head of* $e^{'}$ *is* v_f^- **then**
13 Find a path starting from v_g^+ and ending at v_f^- in P_0;
14 Map each edge of this found path into an arc of D;
15 Add mapped edges not by MM into $M_{(v_g,v_f)}$ **and** Add mapped edges by MM into M_{sub};
16 **return** $M_{(v_g,v_f)}$, M_{sub}; Algorithm terminates;

PROOF Because $M_{(v_g,v_f)} \subseteq E \setminus MM$ in $D = (V, E)$, and when $|M_{(v_g,v_f)}| > 1$, there must be a subset of MM whose each arc shares a common head and a common tail with two different arcs of $M_{(v_g,v_f)}$. Thus, in $B^{'}$, $M_{(v_g,v_f)}$ and M_{sub} together are mapped into a directed path, which starts from v_g^+ and ends at v_f^- in $B^{'}$, which is identified by the second **while** loop.

Therefore, this algorithm mainly identifies $M_{(v_g,v_f)}$ through a directed path of $B^{'}$ that connects v_g^+ and v_f^-. Based on the correctness proof of algorithm 5.1, correctness of this algorithm about identifying such a directed path can be proved as well. ■

Corollary 5.8 (Time complexity of algorithm 5.3)
Except for deriving inputs, by corollary 5.6, time complexity of algorithm 5.3 is $O(|E|)$ in the worst case.

PROOF In detail, except for deriving inputs of this algorithm, running time is cost by mapping D with MM into $B^{'}$ with MM_B based on definition 5.2, and running the path identification based on algorithm 5.1, by which, the worst-case execution time is $O(|E|)$. ■

5.3.4 Time Complexity Analysis

Finally, according to lemma 5.1 and lemma 5.3, a maximum matching of $D \setminus \{u\}$ is identified except for recomputation. This procedure is presented by following algorithm 5.4, where $l = \{0, 1, 2\}$ is used again.

Algorithm 5.4: Derive a maximum matching of $D \setminus \{u\}$.

Input : $D = (V, E)$ with M_D of definition 5.1, u.
Output: A maximum matching of $D \setminus \{u\}$.

```
Identify a maximum matching MM of D by running algorithm 5.2;
if l ∈ {0, 1} then
    Removing u from D;
    return Remaining MM;
    else if l = 2 then
        Derive a matching M_(v_g,v_f) of D by running algorithm 5.3;
        if M_(v_g,v_f) ≠ ∅ then
            Removing u from D;
            return MM \ {{⟨u, v_f⟩, ⟨v_g, u⟩} ∪ M_sub} ∪ M_(v_g,v_f);
            else if then
                Removing u from D;
                return Remaining MM;
```

PROOF According to lemma 5.1, 5.3, to derive a maximum matching of $D \setminus \{u\}$, it needs to identify MM at least. Then, $l \in \{0, 1\}$, remaining MM can be a proposed maximum matching. Further if $l = 2$, $M_{(v_g,v_f)}$ should be identified. After that, a maximum matching of $D \setminus \{u\}$ is obtained, and problem 5.2 is solved. ■

Corollary 5.9 (Time complexity of solving problem 5.2)
Given $D = (V, E)$ with M_D of definition 5.1, the worst-case execution time of solving the problem 5.2 by algorithm 5.4 is $O(|E| \cdot \sqrt{|V|})$.

PROOF For the worst-case execution time, except for identifying M_D of D, and confirming the removed vertex u, it is the sum of time complexity of each procedure,

according to previous algorithm 5.1, 5.2 and 5.3, time complexity of this algorithm is $O(|E|) + O(|E_B|)$. By definition 5.1, due to $|E_B| = |E_|$, the worst-case execution time of solving problem 5.2 is thus $O(|E|)$. When deriving M_D is concerned, with the time complexity $O(|E| \cdot \sqrt{|V|})$ by the algorithm of [60], the worst-case execution time becomes $O(|E| \cdot \sqrt{|V|})$. ■

Furthermore, referring the result of [17], if D is a sparse ER random digraph, the average-case time complexity of solving the problem 5.2 is $O(|E| \cdot log(|V|))$.

5.3.5 Comparison

In the worst case, compared with those existing previous works about structural-control recovery reviewed in section 3.2.1 of chapter 3, our scenario is more efficient than them and excludes assumptions, which are about the types of input networks, tree decomposition and tree width. Additionally, by theorem 5.4, because the maximum matching that contains the minimum number of edges incident to u^+ and u^- are obtained through two augmenting paths at most, so that MM contains the maximum number of matched nodes related to M_D. As a result, unmatched nodes with maximum number that related to M_D in D are still the unmatched nodes related to MM of $D \setminus \{u\}$. Furthermore, it means that previous inputs can be reused to recover structural control into the residual physical system with the maximum number.

5.4 Summary

After removing a known component of an initially minimum-input structurally controllable physical system, this chapter efficiently recovers structural controllability with a minimum set of inputs, so that resilience of the control system containing it can be enhanced. Still based on the maximum-matching based method of section 2.4.1 of chapter 2, we define a digraph with a known maximum matching to represent the state matrix after removing the same indexed single column and row. Then, a maximum matching of this input network is identified, and this maximum matching is identified in linear time except for computing the known maximum matching of the input network. As a result, structural control into residual physical system could be recovered in linear time. After focusing on structural-control recovery against very-limited modification, next chapter recovers structural controllability of a physical system after severe attacks or failures with fixed input matrix.

Chapter 6

Structural-Control Recovery via the Minimum-edge Addition

6.1 Overview

After implementing structural-controllability recovery against very limited modification of system components or network vertices in previous two chapters, it is also indispensable to concern constraints on inputs that are given to structurally control the latest residual system. This is because, after severe attacks or failures on system components, input matrix identified according to the resulting state matrix might require more number of columns or non-zero entries than the input matrix in reality. Reviewing related works about structural-control recovery in section 3.2.1 of chapter 3, none of them mentions the constraints on input matrix. As a result, structural control into residual system may not be effectively recovered in reality, although those methods and scenarios may be executed more efficient than recomputing a power dominating set or a maximum matching. In consequence, given a constrained input matrix, extra modification on the resulting state matrix is undoubtedly essential. Generally, the modification could be replacing zero entries with non-zero entries, or rearranging some existing non-zero entries into other positions. Nevertheless, new single column and single row can not be added to the resulting state matrix, because such addition would require a new input.

Therefore, this chapter solves research question 3. Specifically, with assumptions of section 1.3.2 of chapter 1, there is an initially structurally uncontrollable CT-LTI system and a known input matrix. Then, this chapter efficiently recovers its structural controllability by adding a minimum number of non-zero entries into the residual state matrix. Particularly, the input matrix of this given system is always fixed during recovery.

According to theorem 2.1 and corollary 2.2 in section 2.3 of chapter 2, given a system network that is mapped by both residual state matrix and the given input matrix, we define it as an input network of this chapter. Then, our solution is to add a minimum set of edges into this input network, to eventually construct a digraph spanned by disjoint cacti of definition 2.8, so that the resulting network represents a structurally controllable CT-LTI system, which also involves this initially given input matrix. Additionally, in terms of constructing a structurally controllable system with given inputs, our problem can be alternatively solved by an existing edge-addition scenario of [37], which is illustrated in section 3.2.2 of chapter 3. However, this scenario is implemented with low efficiency. Let m, n be the number of edges and vertices of a system network, the worst-case execution time is $O(n^3)$. By contrast, this chapter would give a different minimum edge-addition scenario with higher efficiency, whose time complexity is just $O(\sqrt{n} \cdot m)$.

To effectively add edges and ensure that the final digraph is spanned by disjoint cacti, given the input network, based on a fixed and arbitrary maximum matching of it, we raise a two-step edge-addition scenario. This scenario detects and removes the dilation of definition 2.6 in the first step, and then removes inaccessible vertices of definition 2.7 finally. According to this scenario, it is further discussed when added edges can be reduced by the most in number. We conclude that the number of added edges in the first step is a constant value for the given system network, while the number of added edges in the second step varies. It is thus possible to achieve the minimum-edge addition for the given system network. Eventually, our scenario returns a digraph spanned by a set of disjoint cacti, and the time complexity is the same as that of identifying a maximum matching of the input network.

For the contribution of this chapter, given a structurally uncontrollable system with a fixed input matrix, in order to recover structural controllability, the worst-case execution time of applying minimum-edge addition is $O(\sqrt{n} \cdot m)$.

This chapter is structured as follows: section 6.2 specifies research question; section 6.3 constructs disjoint cacti, and the last section 6.4 summarizes this chapter.

6.2 Problem Formulation

Given a CT-LTI system, which was structurally controllable, after severe attack or failure, its residual system is now structurally uncontrollable with a given input matrix, and this structurally uncontrollable system is described by a state equation below:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{6.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times m}$. Particularly, matrix $\mathbf{B}$ is fixed, which means that any entries of each column of $\mathbf{B}$ can not be changed. Then, research question of this chapter is formulated:

Research Question: Identify a matrix with the minimum number of non-zero entries, noted by $\mathbf{A}^{'} \in \mathbb{R}^{n\times n}$, so that the resulting system described by following equation is structurally controllable:

$$\dot{x}(t) = (\mathbf{A} + \mathbf{A}^{'})x(t) + \mathbf{B}u(t) \tag{6.2}$$

By corollary 2.2 and definition 2.4 of chapter 2, the input network of this chapter is defined in definition 6.1, which is assumed to be the system network mapped by matrix $\mathbf{A}$ and matrix $\mathbf{B}$ of equation (6.2).

Definition 6.1 (Input Network of chapter 6)
Let $D = (V \cup U, E)$ be a large and finite digraph, which excludes self loops, isolated vertices and parallel arcs. Also, let $V = \{v_i | 1 < i \leq n\}$ and $U = \{u_r | 1 \leq r \leq m\}(m < n)$ be two independent vertex sets, where each node of U has no in degree and it is a tail of the arc whose head is only a node of V. Besides, let E be a set of edges among vertices of $V \cup U$, and M_D be a fixed and arbitrary maximum matching of D.

For each arc of E, it corresponds to only one non-zero entry of either $\mathbf{A}$ or $\mathbf{B}$ of equation (6.1). With $D = (V \cup U, E)$ of definition 6.1, the research question could be transferred into the following graph-theoretical problem:

Problem 1: Given digraph $D = (V \cup U, E)$ of definition 6.1. Then, adding a minimum set of edges only among different nodes of V into D, so that the resulting digraph is spanned by a set of disjoint cacti.

Problem 6.2 is eventually solved in following sections as a way to solve the research question from a graph-theoretical aspect. For the solution, it mainly relies on the maximum matching of D to identify and construct disjoint cacti.

6.3 Disjoint Cacti Construction

In this section, given $D = (V \cup U, E)$ of definition 6.1, we show a scenario of constructing a graph spanned by disjoint cacti in subsection 6.3.1 and 6.3.2. Based on this scenario, in subsection 6.3.3, we confirm the minimum number of added edges into D, and the related algorithms are shown in section 6.3.4 finally.

6.3.1 The first edge-addition step

By theorem 2.1 and corollary 2.2 of chapter 2, we already known if a system network represents a structurally controllable system, it must exclude both the dilation and inaccessible vertices, and this network work should also be spanned by disjoint cacti. We thus use the dilation of D as a clue to guide our edge addition in the first step. We conclude lemma 6.1 and corollary 6.2 to detect the dilation of $D = (V \cup U, E)$ of definition 6.1, by which, we then conclude corollary 6.3 to further guide and justify the first edge-addition step, which is shown by algorithm 6.1.

Lemma 6.1
Given $D = (V \cup U, E)$ with M_D of definition 6.1. Then, with respect to M_D, if D contains the dilation, the number of unmatched nodes that are heads of arcs of E is more than that of unmatched nodes that are tails of arcs of E.

PROOF Let $V(M_D)$ be a set of vertices of V incident to arcs of M_D.

Firstly, focusing on the digraph: $M_D \cup U \cup \{V \setminus V(M_D)\}$, which contains isolated vertices, vertex-disjoint directed paths or cycles. Then, all heads of arcs of $M_D \cup U \cup \{V \setminus V(M_D)\}$ and all tails of arcs of $M_D \cup U \cup \{V \setminus V(M_D)\}$ are only from directed paths and cycles of M_D, so that the number of them are same. Besides, since $M_D \cup U \cup \{V \setminus V(M_D)\}$ contains all vertices of $V \cup U$, $M_D \cup U \cup \{V \setminus V(M_D)\}$ thus spans D. Further, when any arc of $E \setminus M_D$ is added into $M_D \cup U \cup \{V \setminus V(M_D)\}$, the added arc can either increase the number of tails, or increase that of heads, while the newly-increased heads and tails can not be produced at the same time. Otherwise, maximality of M_D of D is contradicted. Moreover, with respect to M_D, during adding edges of $E \setminus M_D$ into $M_D \cup U \cup \{V \setminus V(M_D)\}$, new tails are produced by isolated unmatched nodes related to M_D and endings nodes of directed paths of M_D, while new heads are only produced by unmatched nodes of V. Obviously, within D and related to M_D, these unmatched nodes are either unmatched nodes that are tails of arcs of E, or unmatched nodes that are heads of arcs of E.

Secondly, if D involves the dilation, according to definition 2.6, the number of heads of arcs of E must be more than that of tails of arcs of E, and it means that the number of newly produced heads must be more than that of newly produced tails, during adding edges of $E \setminus M_D$ into $M_D \cup U \cup \{V \setminus V(M_D)\}$. Above all, in

comparison of the sources of newly generated tails and heads during adding edges of $E \setminus M_D$ into $M_D \cup U \cup \{V \setminus V(M_D)\}$, with respect to M_D, unmatched nodes as heads of arcs of E must be more than that of both unmatched nodes as tails of arcs of E and ending nodes of directed paths of M_D. Because ending nodes of directed paths of M_D are matched nodes related to M_D, if D involves the dilation, it is true that the number of unmatched nodes that are heads of arcs of E must be more than that of unmatched nodes that are tails of arcs of E. ■

According to lemma 6.1, corollary 6.2 is concluded to indicate when a digraph like D can exclude the dilation.

Corollary 6.2
Given $D = (V \cup U, E)$ with M_D of definition 6.1. If the number of unmatched nodes that are heads of arcs of E is less than or equal to that of unmatched nodes that are tails of arcs of E, then D excludes the dilation.

PROOF Given digraph $M_D \cup U \cup \{V \setminus V(M_D)\}$ of lemma 6.1, and based on the source of newly-generated heads and tails during adding arcs of $E \setminus M_D$ into it, when the number of unmatched nodes that are heads of arcs of E is less than or equal to that of unmatched nodes that are tails of arcs of E, the number of tails of arcs of E must be more than or equal to that of heads of arcs of E. By definition 2.6, D excludes the dilation. ■

After that, according to corollary 6.2, following corollary 6.3 determines the first edge-addition step, which is then clearly illustrated by following algorithm 6.1.

Corollary 6.3
Given $D = (V \cup U, E)$ with M_D of definition 6.1, and D contains the dilation, let E_{a_1} be a set of added arcs into D to remove the dilation of D, where arcs of E_{a_1} are only incident to nodes of V. Then, if each node of V is a matched node related to a maximum matching $\{M_D \cup E_{a_1}\}$, by corollary 6.2, digraph $(V \cup U, E \cup E_{a_1})$ excludes the dilation.

PROOF With respect to maximum matching $M_D \cup E_{a_1}$, if all nodes of V are matched. As a result, unmatched nodes that are heads of arcs of $E \cup E_{a_1}$ do not exist in $(V \cup U, E \cup E_{a_1})$. By contrast, unmatched nodes of $(V \cup U, E \cup E_{a_1})$ related to $M_D \cup E_{a_1}$ are only nodes of U, and they are always tails of arcs of $(V \cup U, E \cup E_{a_1})$. After that, it is obvious that the number of unmatched nodes that are heads of arcs of $E \cup E_{a_1}$ is definitely less than that of unmatched nodes that are tails of arcs of $E \cup E_{a_1}$. By corollary 6.2, digraph $(V \cup U, E \cup E_{a_1})$ must exclude the dilation. ■

Based on corollary 6.3, the first edge-addition step is shown in algorithm 6.1. Here, let $V_{M_D} \subseteq V$ be a set of unmatched nodes of V and related to M_D. And let each node of V_{M_D} be v_i, and E_{a_1} be an initially empty arc set, which would collect all added arcs among vertices of V. Besides, let $e_i \notin E$ be a single added arc, where head of e_i is v_i, and tail of e_i is an ending vertex of a path of the currently identified maximum matching.

PROOF This algorithm firstly identifies a maximum matching M_D of D in order to remove the dilation of D by corollary 6.2, if there is no unmatched node of V related to M_D, D excludes the dilation and $E_{a_1} = \emptyset$. Otherwise, related to M_D,

Algorithm 6.1: The first edge-addition step.

Input : $D = (V \cup U, E)$ of definition 6.1, E_{a_1}.
Output: A digraph excluding the dilation.

```
Identify a maximum matching M_D of D;
Identify V_{M_D} through each starting node of directed paths of M_D and
  vertices not incident to any edge of M_D;
while |V_{M_D}| ≠ ∅ and v_i ∈ V_{M_D} do
    V'_{M_D} = V_{M_D} \ v_i;
    E'_{a_1} = E_{a_1} ∪ e_i;
return (V ∪ U, E ∪ E_{a_1}), M_D;
```

this procedure adds a set of edges to make all nodes of V be matched vertices with respect to a maximum matching of the resulting digraph. Also, by definition 6.1 and corollary 6.3, since any vertex of U can never be a matched node related to any matching of D or D after adding edges, cardinality of the maximum matching of resulting digraph obtained by adding edges into D can be $|V|$ at most. Further, in the **while** loop, after adding an arc, it is obvious that remaining unmatched nodes of V and related to M_D are still unmatched related to the latest identified maximum matching, which are then adjacent to the ending node of a directed path of the latest identified matching by following edge addition. Due to removal of each node of V_{M_D}, **while** loop terminates at $V_{M_D} = \emptyset$. Until there is no unmatched node of V related to M_D, the resulting digraph is returned. ■

Corollary 6.4 (Time complexity of algorithm 6.1)
Except for deriving $D = (V \cup U, E)$ of definition 6.1, the worst-case execution time of running algorithm 6.1 is $O(\sqrt{|V \cup U|}|E|)$.

PROOF For the time complexity of this procedure, it is the sum running time of identifying a maximum matching M_D, identifying unmatched nodes of V and related to M_D, and running the **while** loop. By the algorithm of [60], identifying M_D costs $O(\sqrt{|V \cup U|}|E|)$ steps at most. For unmatched nodes of V and related to M_D, they are identified by scanning each starting node of directed paths of M_D and nodes of V that are not heads of arcs of M_D, in $O(|V|)$ steps at most. Also, running the **while** loop costs $O(|V|)$ steps. In total, worst-case execution time complexity of the first edge-addition step is $O(\sqrt{|V \cup U|}|E|)$. ■

One more thing, by lemma 6.1 and corollary 6.2, although purely removing the dilation of D may not require to make all unmatched nodes of V related to M_D be matched with respect to a maximum matching of the resulting digraph, it is clearly that the first edge-addition step not only removes the dilation of D, but also make any single nodes of all directed paths of $M_D \cup E_{a_1}$ be accessible from vertices of U. Further, given $(V \cup U, E \cup E_{a_1})$ returned by algorithm 6.1, adding more edges into it can not introduce the dilation, this is because any node of V has been a matched node, or a head of an arc.

Simultaneously, since the number of unmatched nodes with respect to any maximum matching of finite digraph $D = (V \cup U, E)$ is a constant value, $|E_{a_1}| =$

$|V| - |M_D|$ is thus constant according to D, and our first edge-addition step requires a constant number of added edges.

6.3.2 The second edge-addition step

After implementing the first edge-addition step for $D = (V \cup U, E)$ of definition 6.1, we now detect vertices of V that are inaccessible from nodes of U in digraph $(V \cup U, E \cup E_{a_1})$. To do this, we conclude lemma 6.5 based on a kind of strongly connected components, which is defined by definition 6.2. Generally, a strongly connected component of a digraph is a subgraph whose any pair of vertices are connected through at least one existing directed path.

Definition 6.2 (S_{cc})
Given $D = (V \cup U, E)$ with M_D of definition 6.1 and E_{a_1} returned by algorithm 6.1. Then, let S_{cc} be a set of all strongly connected components that are only spanned by one or more cycles of M_D, and exclude vertices as heads of arcs whose tails are out of S_{cc} in $D \cup E_{a_1}$.

Lemma 6.5
Given $D = (V \cup U, E)$ with M_D of definition 6.1 and E_{a_1} of algorithm 6.1. Then, digraph $(V \cup U, E \cup E_{a_1})$ contains vertices of V that are inaccessible from nodes of U, if and only if $S_{cc} \neq \emptyset$.

PROOF $\Leftarrow$: Within $(V \cup U, E \cup E_{a_1})$, if there are strongly connected components that only involve one or more cycles of $M_D \cup E_{a_1}$, and exclude nodes pointed by vertices out of them. Obviously, vertices of $S_{cc} \neq \emptyset$ can not be approached by nodes out of these components through existing paths. By contrast, after implementing algorithm 6.1, all vertices of V that are out of cycles of M_D must be approachable from nodes of U. This is because $(V \cup U, E \cup E_{a_1})$ has no unmatched nodes that are contained by V, where each node of V is either in a path starting from a node of U or in the cycle of M_D, and any cycle of M_D excludes unmatched nodes related to M_D. As a result, nodes of S_{cc} are only inaccessible vertices of V from nodes of U in $(V \cup U, E \cup E_{a_1})$.

$\Rightarrow$: If $(V \cup U, E \cup E_{a_1})$ contains inaccessible vertices of V from U. As mentioned above, inaccessible nodes of V from nodes of U in $(V \cup U, E \cup E_{a_1})$ are only contained by inaccessible cycles of M_D by nodes of U. Let c_i be an arbitrary inaccessible cycle of M_D from nodes of U, then, c_i could have no vertices as heads of arcs whose tails are out of c_i, so that nodes of U can not visit c_i through existing paths on the one hand. On the other hand, cycles of M_D, which are connected with c_i in $(V \cup U, E \cup E_{a_1})$, must be also inaccessible from nodes of U in $(V \cup U, E \cup E_{a_1})$. Otherwise, c_i can be accessible from nodes of U. Further, either those single c_i or cycles of M_D that are connected with c_i are all strongly connected component of finite digraph $(V \cup U, E \cup E_{a_1})$. Therefore, when $(V \cup U, E \cup E_{a_1})$ contains inaccessible nodes of V from U, there are strongly connected components only involving one or more disjoint cycles of M_D and excluding nodes pointed by vertices out of them. ■

After that, if $(V \cup U, E \cup E_{a_1})$ contains the inaccessible nodes of V from U, corollary 6.6 is concluded to remove them by adding extra arcs.

Corollary 6.6
Given $(V \cup U, E \cup E_{a_1})$ returned by algorithm 6.1, let E_{a_2} be a set of added arcs into it. Then, according to lemma 6.5, if $S_{cc} = \emptyset$ in $(V \cup U, \{E \cup E_{a_1}\} \cup E_{a_2})$, then, inaccessible vertices of V from nodes of U are excluded by $(V \cup U, \{E \cup E_{a_1}\} \cup E_{a_2})$.

PROOF According to the proof of lemma 6.5, since inaccessible vertices of V from U can only exist in cycles of $M_D \cup E_{a_1}$. Then, if $S_{cc} = \emptyset$ in $(V \cup U, \{E \cup E_{a_1}\} \cup E_{a_2})$, no inaccessible cycles from U can exist, and inaccessible vertices of V from U in $(V \cup U, E \cup E_{a_1})$ can thus not exist as a result. ■

Based on corollary 6.6, the second edge-addition step is shown in algorithm 6.2 below. Here, let E_{a_2} be an initially empty set, and e_j be an added arc into $(V \cup U, E \cup E_{a_1})$. Also, let s_i be any element of S_{cc}, and head of e_j be a vertex of s_i and tail is a node of a directed path of $M_D \cup E_{a_1}$.

Algorithm 6.2: The second edge-addition step.

Input : $(V \cup U, E \cup E_{a_1})$ and M_D returned by algorithm 6.1.
Output: A digraph without the dilation and inaccessible nodes.

```
1 Identify S_cc through cycles of M_D;
2 while S_cc ≠ ∅ and s_i ∈ S_cc do
3 |   S'_cc = S_cc \ s_i;
4 |   E'_a2 = E_a2 ∪ e_j;
5 return (V ∪ U, {E ∪ E_a1} ∪ E_a2);
```

PROOF With $(V \cup U, E \cup E_{a_1})$ and M_D returned by algorithm 6.1, S_{cc} is identified by scanning nodes of cycles of M_D based on the depth-first search or breath-first search [42] in $O(|V| + |E \cup E_{a_1}|)$ steps at most. Then, for each element of S_{cc}, an arc e_j is added to make it be accessible from nodes of U. Due to line 3, when $S_{cc} = \emptyset$, the second edge-addition step terminates and the digraph $(V \cup U, \{E \cup E_{a_1}\} \cup E_{a_2})$ excludes both inaccessible nodes of V from U and the dilation. ■

Corollary 6.7 (Time complexity of algorithm 6.2)
Given digraph $(V \cup U, E \cup E_{a_1})$ returned by algorithm 6.1, the worst-case execution time of running algorithm 6.2 is $O(|V| + |E \cup E_{a_1}|)$.

PROOF For the worst-case execution time, it is cost by identifying S_{cc} and running the **while** loop. Thus, with proof of algorithm 6.2, except for running algorithm 6.1 to obtain $(V \cup U, E \cup E_{a_1})$, it is $O(|V| + |E \cup E_{a_1}|)$, this is also because $|S_{cc}| < |V|$.■

According to corollary 6.3 and corollary 6.6, it is clear that $E_{a_1} \cap E_{a_2} = \emptyset$ and the edge-addition scenario is further concluded by theorem 6.8 below:

Theorem 6.8 (Edge-addition Scenario)
Given $D = (V \cup U, E)$ of definition 6.1, then, after adding arcs by corollary 6.3 and 6.6 in order, the digraph $(V \cup U, \{E \cup E_{a_1}\} \cup E_{a_2})$ is spanned by a set of vertex-disjoint cacti.

PROOF Firstly, by corollary 6.3, the dilation of D is removed by adding a set of arcs E_{a_1}, and all current directed paths of $M_D \cup E_{a_1}$ must start from nodes of U. Then, because $\{M_D \cup U\} \cup E_{a_1}$ spans $(V \cup U, E \cup E_{a_1})$, and cycles of $M_D \cup E_{a_1}$ in $(V \cup U, E \cup E_{a_1})$ might already have constructed a set of disjoint buds of definition 2.5 with edges incident to vertices of them. After adding E_{a_2} into $(V \cup U, E \cup E_{a_1})$ by corollary 6.6, those previously inaccessible disjoint buds that are span S_{cc} can be accessible by nodes of U through paths of $M_D \cup E_{a_1}$, so that all cycles of M_D are involved into buds. Eventually, $(V \cup U, \{E \cup E_{a_1}\} \cup E_{a_2})$ is spanned by a set of disjoint cacti, where each cactus starts from a node of U. ■

By theorem 6.8, the number of added arcs is represented by $|E_{a_1}| + |E_{a_2}|$. Besides, reviewing algorithm 6.1, because all matched nodes of D and related to M_D can be only contained by V. Thus, $|V_{M_D}| = |V| - |M_D|$. Also, since $|V_{M_D}| = |E_{a_1}|$, and $|M_D|$ is a constant value for D. Therefore, $|E_{a_1}|$ is a constant number according to the given digraph D. Then, next section explores when $|E_{a_2}|$ can be reduced by the most in order to obtain the minimum set of added edges.

6.3.3 The Minimum Number of Added Arcs

Firstly, we conclude theorem 6.9 to indicate how to reduce the number of added edges by one:

Theorem 6.9
Given $D = (V \cup U, E)$ with M_D of definition 6.1, and $S_{cc} \neq \emptyset$, let $\overrightarrow{\langle v_i, v_j \rangle}$ be an arc of a cycle of an element of S_{cc}, and $v_k \in V$ be an unmatched node related to M_D. Then, according to M_D, by the edge-addition scenario of theorem 6.8, the total number of added edges into D is reduced by one, if and only if the arc $\overrightarrow{\langle v_i, v_k \rangle} \in \{E \setminus M_D\}$ exists, and v_j is an unmatched node related to a different maximum matching from M_D.

PROOF Let $M_D^{'}$ be a maximum matching of D, and $M_D^{'} \neq M_D$. Since the number of edges added in the first step is constant, we thus prove that the number of added edges in the second step according to $M_D^{'}$ is less than that according to M_D by one, if and only if arc $\overrightarrow{\langle v_i, v_k \rangle}$ exists, and v_j is an unmatched node related to $M_D^{'}$.

$\Leftarrow$: If arc $\overrightarrow{\langle v_i, v_k \rangle} \in \{E \setminus M_D\}$ exists, and v_j is an unmatched node related to $M_D^{'}$. Then, in $M_D^{'}$, a path can exist, which starts from v_j, contains all vertices of a cycle of M_D and involving $\overrightarrow{\langle v_i, v_j \rangle}$, and also includes a path of M_D starting from v_k. An example is illustrated in figure 6.1. In this example, the red cycle and the red path are contained into a same maximum matching, and this cycle is an element of S_{cc}. Also, due to the existence of the arc $\overrightarrow{\langle v_4, v_6 \rangle}$, a directed path starting from v_5 and ending at v_{10} exists, and it is excluded by the maximum matching containing this red cycle and path.

After the first edge-addition step by $M_D^{'}$, we can observe that all nodes of an element of S_{cc}, which includes $\overrightarrow{\langle v_i, v_j \rangle}$, is now accessible from nodes of U, and there is no need for extra edges to make this element of S_{cc} be accessible again by corollary 6.6. Nevertheless, by M_D, after the dilation removal step, such same element of S_{cc} is still inaccessible from nodes of U, which thus requires an edge to make nodes of it accessible. Hence, the total number of added edges into D according to $M_D^{'}$ is less than that according to M_D by one.

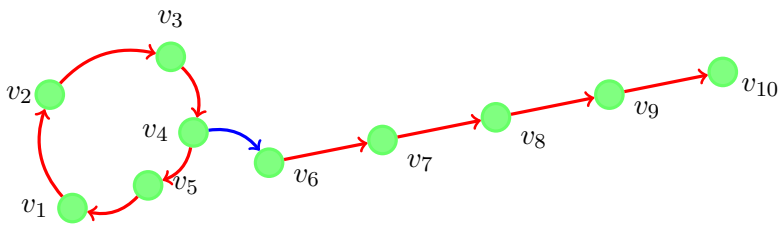

Figure 6.1:

$\Rightarrow$: After the first edge-addition step, based on corollary 6.6, if the number of added edges to eliminate nodes of V that are inaccessible from U according to $M_D^{'}$ is less than that according to M_D by one. Let $S_{cc}^{'}$ be a set of all strongly connected components that only involve one or more disjoint cycles of $M_D^{'}$, and exclude nodes pointed by vertices out of them after removing dilation of D. Then, because adding edges according to any maximum matching of D in the first edge-addition step does not influence any cycle of this maximum matching. The number of elements of S_{cc} is thus more than that of $S_{cc}^{'}$ by one, which can be represented by $|S_{cc}^{'}| = |S_{cc}|-1$. Also, since any two maximum matchings of a same digraph can be transformed into each other by exchanging vertices or edges, it is possible that one element of S_{cc} should be out of both S_{cc} and $S_{cc}^{'}$ by some ways. Based on definition 6.2, for this purpose, it requires that all nodes of a cycle of this element of S_{cc} must be contained into a path of $M_D^{'}$. In detail, for the element of S_{cc} only involving a single cycle of M_D, we should make its vertices be contained by a path and without changing the number of unmatched nodes of D. Besides, for the element of S_{cc} involving multiple disjoint cycles of M_D, we can make any involved single cycle's vertices be contained by a path and without changing the number of unmatched nodes of D, so that other cycles of this element would be accessible. Above all, given M_D, nodes of a path of $M_D^{'}$ that are all vertices of a cycle and a path together of M_D, in which the starting vertex of this path is noted by v_k, and an edge of this cycle is noted by $\overrightarrow{\langle v_i, v_j \rangle}$. Then, there must be $\overrightarrow{\langle v_i, v_k \rangle} \in M_D^{'}$, and v_j is the starting vertex of this path of $M_D^{'}$. ■

According to theorem 6.9, corollary 6.10 is concluded to reduce the maximum number of added edges based on the edge-addition scenario of theorem 6.8:

Corollary 6.10
Given $D = (V \cup U, E)$ with M_D of definition 6.1, $S_{cc} \neq \emptyset$, and let S_{sub} be a subset of S_{cc}. Also, assume that each element of S_{sub} have a vertex as the tail of an arc whose head is an unmatched node of V related to M_D. Then, if cardinality of S_{sub} is maximum, the total number of added edges into D according to M_D is reduced by the most by theorem 6.9.

PROOF Given any element of S_{sub}, based on theorem 6.9, the number of added edges into D according to M_D can be reduced by one. Also, because elements of S_{cc} are mutually disjoint components, once a vertex of any element of S_{sub} is discovered, which is an unmatched node with respect to a maximum matching different from M_D, there can be a common maximum matching different from M_D for all such vertices. Besides, since D is a finite graph, $S_{sub} \subseteq S_{cc}$ is a finite strongly

connected component. Therefore, in aggregation, when cardinality of S_{sub} is the maximum, the number of added edges by scenario of theorem 6.8 is reduced by the most. ■

6.3.4 Execution

According to corollary 6.3, 6.6 and corollary 6.10, algorithm 6.3 shows the entire process of constructing a digraph spanned by a set of disjoint cacti via adding a minimum set of edges into $D = (V \cup U, E)$ through M_D of definition 6.1. Here, let $v_h \in V$ be any vertex not incident to edges of M_D. Besides, let s_i be an element of S_{cc}, p_i be a path of M_D, and G be an initially empty set. Also, let $v_k \in V$ be either the starting node of p_i, or an isolated unmatched node related to M_D, and let c_i be a cycle, $\overrightarrow{\langle v_i, v_j \rangle} \in E$ be an arc of c_i.

Algorithm 6.3: Construct a digraph spanned by disjoint cacti

Input: $D = (V \cup U, E)$ of definition 6.1, G
Output: A digraph spanned by disjoint cacti

1 Starting from each node of U, find M_D by running the algorithm of [60];
2 Identify each c_i, p_i of M_D by depth-first search (DFS) algorithm of [111], and each v_h;
3 Add each v_h into G;
4 Identify S_{cc} from each c_i in D; $G' = G \cup S_{cc}$;
5 Remove cycles of S_{cc} from M_D;
6 Identify each $s_i \in S_{cc}$, for $c_i \in s_i$, where $\overrightarrow{\langle v_i, v_j \rangle} \in c_i$, and $\overrightarrow{\langle v_i, v_k \rangle} \in E \setminus M_D$;
7 **for** *each identified* s_i **do**
8 $\quad G' = G \setminus c_i$ **and** $S'_{cc} = S_{cc} \setminus s_i$;
9 $\quad$ **if** v_k *is an isolated unmatched node related to* M_D **then**
10 $\quad\quad G' = G \setminus v_k$;
11 $\quad\quad$ Construct a path: $\{c_i \setminus \overrightarrow{\langle v_i, v_j \rangle}\} \cup \overrightarrow{\langle v_i, v_k \rangle}$;
12 $\quad\quad$ **else if then**
13 $\quad\quad\quad$ Construct a path: $\{c_i \setminus \overrightarrow{\langle v_i, v_j \rangle}\} \cup \{p_i \cup \overrightarrow{\langle v_i, v_k \rangle}\}$ **and** $M'_D = M_D \setminus p_i$;
14 $\quad$ Add this path into G;
15 $G' = G \cup M_D$;
16 In G, add arcs into G from each existing zero-outdegree node of V to each zero-indegree node of V until there is no zero-indegree node of V ;
17 In G, add arcs into G from any node of V and out of S_{cc} to each element of S_{cc} until each element of S_{cc} has an incoming edge whose tails is out of S_{cc};
18 **return** $D \cup G$;

PROOF Initially, this procedure identifies a maximum matching M_D to detect the dilation and inaccessible nodes of D in line 1. Particularly, M_D is identified from edges incident to vertices of U in order to sufficiently use vertices of U. Then, line 2-4 identifies S_{cc} according to cycles of M_D, and S_{cc} is added into G. In procedure of line 5, cycles of S_{cc} are removed from M_D, which results in that current M_D involves disjoint paths and cycles, and those cycles must contain vertices pointed

by nodes out of them. By theorem 6.9 and corollary 6.10, procedures of line 6 and 7 modifies some strongly connected components of S_{cc} to reduce the number of added edges according to M_D by the most. Because D is finite, the number of elements of S_{cc} is finite, and each chosen s_i is removed in line 8, this **for** loop can terminate. During the modification, paths obtained through procedure of either line 9 or 11 are added into G at line 14. After running **for** loop, current M_D only involves disjoint paths and cycles out of S_{cc}, while current G only contains isolated single vertices and disjoint paths and remaining elements of S_{cc}. Next, M_D is add into G in line 15, and G thus contains disjoint paths, single vertices not incident to M_D, and remaining S_{cc} whose all elements have no incoming edges from nodes out of them, which together span original D, and each of them is disjoint with others. Also, there is no common vertices or edges within G. Clearly, in G, all vertices of V without indegree are unmatched nodes related to a common maximum matching different from M_D. Later, the first edge-addition step is executed by line 16, which adds edges to remove all unmatched nodes of V and related to a common maximum matching that is different from M_D, so that the resulting digraph has no dilation and all paths starting from nodes of U. In the following, line 17 removes inaccessible nodes of V from U. Specifically, those inaccessible nodes are removed through vertices of remaining elements of S_{cc} in G. After that, G contains all vertices of D, and all remaining elements of S_{cc} are adjacent with disjoint paths that start from U. Thus, a set of disjoint cacti that start from nodes of U must exist in G. Eventually, $D \cup G$ is therefore spanned by a set of disjoint cacti that only start from nodes of U.

Corollary 6.11 (Time complexity of algorithm 6.3)
Given $D = (V \cup U, E)$ of definition 6.1, the worst-case execution time of running algorithm 6.3 or solving the problem 6.2 is $O(\sqrt{|V \cup U|} \cdot |E|)$.

PROOF For the worst-case execution time of this algorithm, it is the sum of running time of each line. Identifying a maximum matching of D costs $O(\sqrt{|V \cup U|} \cdot |E|)$ by algorithm [60], and identifying cycles and paths of M_D cost $O(|V \cup U| + |E|)$ through executing DFS algorithm. Then, for running line 4, it needs to traverse each identified cycles of M_D. The worst-case time complexity is thus represented by $O(|V|+|E|)$ at most. In detail, treat each cycle as a vertices, and edges among those cycles are still edges. Then, by using DFS algorithm again, components that only composed of cycles of M_D can be identified and S_{cc} can be obtained, which only contains components without incoming edges of E. Meanwhile, for each identified component, the involved cycles are directly removed from M_D in line 5 rather than identifying them again. Next, for running procedure of line 6, it needs to traverse each node of an element of S_{cc} at most, which thus cost $O(|V|)$ steps at most. For running the **for** loop of line 7, since combining nodes of each s_i with an unmatched node related M_D into a path of $E \setminus M_D$ costs $O(1)$, because each c_i and paired single node or path has been known. The worst-case execution time of this procedure is $O(|E|)$. Later, adding edges of the first step in line 16 needs to identify zero-indegree and zero-outdegree nodes, which can be done in $O(|V|)$ steps at most by traversing each node. As for the second edge-addition step of line 17, because remaining elements of S_{cc} in G has been already known after the **for** loop, the edge

addition only depends on the existence of elements of S_{cc}, which thus costs $O(|E|)$ steps at most. Above all, worst-case execution time of algorithm 6.3 is $O(\sqrt{|V \cup U|} \cdot |E|)$ for $D = (V \cup U, E)$. ■

6.3.5 Comparison

Compared with the edge-addition scenario of [37], which is specifically reviewed in section 3.2.2 of chapter 3, the input network of both this scenario and our input network of definition 6.1 are same. Nevertheless, for the time complexity of executing our entire operations in the worst case, it is equivalent to that of identifying a maximum matching of the system network, and more efficiently than running the scenario of [37], which is proportional to the cube of the number of vertices of the input network.

Additionally, our scenario might be executed more efficient, if $D = (V \cup U, E)$ is a sparse ER random digraph. Because the average-case time complexity of solving the problem 6.2 is reduced into $O(|E| \cdot log(|V \cup U|))$.

6.4 Summary

Given a structurally uncontrollable system, recovery of its structural controllability can be done by various methods and requirements. In this chapter, our recovery is constrained by inputs and time complexity, where the input matrix is fixed. For our solution, we add a minimum set of edges into the given system network to obtain a digraph spanned by a set of disjoint cacti, so that the system represented by this digraph is structurally controllable. Also, the worst-case execution time is more efficient than existing related works.

Part III

Efficient Network Analysis to Maintain Structural Controllability

Chapter 7

Security-Aware Edge Analysis for Structural Controllability

7.1 Overview

Related works of section 3.3 of chapter 3 show the robustness of network structural controllability. And previous works of section 3.4 indicate how single node and edge obtain network structural controllability with a minimum set of inputs. However, it is still unknown how each single node and edge quantitatively maintain structural control with a minimum set of inputs. As a result, vulnerable single nodes and edges to the removal can not be explicitly identified, and let along protecting network structural controllability against removing them. From this chapter to chapter 9, efficient network edge and vertex analysis would be systematically illustrated.

Given a continuous-time and linear time-invariant (CT-LTI) dynamical network like $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4 of chapter 2, to clarify the importance of a single edge in maintaining its structural controllability with a minimum set of inputs, Liu $et\ al.$ [73] defined critical, redundant, and ordinary categories. In detail, a removal of a critical edge gains the minimum number of inputs to structurally control the residual network; removing a redundant edge never affects current minimum set of inputs; removing an ordinary link does not change the minimum number of inputs, except for driver nodes forced by inputs. Obviously, exactly knowing edges of each category is forward-looking to protect structural controllability of a given network against the single edge removal. Yet, an efficient network analysis to confirm categories of all edges of a CT-LTI model network is still uncertain, except for using the low-efficiency algorithm of [97]. Generally speaking, given a digraph with n vertices and m edge, time complexity of this algorithm is $O(m^2 \cdot \sqrt{n} + \sqrt{n} \cdot m)$ in the worst case.

Therefore, this chapter solves research question 4. Specifically, with assumptions of section 1.3.2 of chapter 1, this chapter efficiently classifies all arcs of a minimum-input structurally controllable digraph into critical, redundant and ordinary categories, respectively. Besides, the minimum set of inputs is assumed to be known and obtained by the maximum-matching based method shown in section 2.4.1 of chapter 2. Meanwhile, this digraph is also assumed to contain a precomputed maximum matching.

This research question is solved by identifying all single arcs that are contained by at least one maximum matching of the input network. By theorem 2.8 of chapter 2, this is because a maximum matching of a digraph determines a minimum set of inputs to structurally control itself. Category of any single edge of the input network can be confirmed by analysing if it is involved into a maximum matching

of the given network. Specifically, in this chapter, it would be explained when an edge is out of any maximum matching, it is a redundant edge. When an edge is in some maximum matchings, it is an ordinary edge. When an edge is in all maximum matchings, it is a critical edge. Simultaneously, given a known maximum matching of the input network, although algorithms of [112] can identify some edges of maximum matchings in linear time, they might not be able to identify all such edges. Thus, this chapter is highly motivated to design few algorithms to ensure that each edge involved into a maximum matching can be more explicitly identified in linear time. For this purpose, the input network is mapped into a bipartite graph to run various operations.

For the contribution of this chapter, the worst-case execution time of classifying all edges of a minimum-input structurally controllable network is linear. This is achieved by efficiently identifying all arcs contained by maximum matchings of the input network.

Remaining chapter is structured as follows: section 7.2 specifies research question with some statements; section 7.3 implements the entire arc identification, and the last section 7.4 summarizes this chapter.

7.2 Preliminaries & Problem Formulation

7.2.1 Research Question

Firstly, the input network of this chapter is defined below:

Definition 7.1 (Input Network of chapter 7)
Let $D = (V, E)$ be a large, finite digraph, and D excludes self loops, parallel arcs and isolated nodes. Also, V represents the vertex set, $V = \{v_i | 1 \leq i \leq n\}(n > 2)$, and E represent the edge set, $E = \{\overrightarrow{\langle v_i, v_j \rangle} | v_i, v_j \in V\}$. Besides, let M_D be a fixed and arbitrary maximum matching of D, identified by the algorithm of [60].

Then, assume that this input network and a minimum set of inputs together construct a structurally controllable system, which is also represented by following state equation:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{7.1}$$

where, $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, and $D = (V, E)$ of definition 7.1 only maps into the matrix $\mathbf{A}$. For each arc $\overrightarrow{\langle v_i, v_j \rangle} \in E$, it corresponds to only one non-zero entry, which is noted by $a_{ji} \in \mathbf{A}$. Also, the number of columns of matrix $\mathbf{B}$ is minimum. With digraph $D = (V, E)$ and precomputed M_D, research question of this chapter is specified:

Research Question: Given digraph $D = (V, E)$ of definition 7.1, then, efficiently classify all arcs of E into critical, redundant, and ordinary categories, respectively.

7.2.2 Modelling

In order to model this research question into following graph-theoretical problem 7.2.2, theorem 7.1 and corollary 7.2 are deduced.

Specifically, in the remaining of this chapter, we define M_0 as an another arbitrary maximum matching of D, where $M_0 \neq M_D$ and $|M_0| = |M_D|$. Then, let e be an arbitrary single edge of E, and all possible impacts of removing e on the maximum matching of $D \setminus e$ is shown in the first place:

Theorem 7.1
Given $D = (V, E)$ with M_D of definition 7.1, and M_0. Then, on the one hand, if $e \notin M_D$, M_D is still a maximum matching of digraph $(V, E \setminus e)$; on the other hand, if $e \in M_D$, and $e \notin M_0$, M_0 is still a maximum matching of digraph $(V, E \setminus e)$; besides, if $e \in M_D$ and $e \in M_0$, a maximum matching of digraph $(V, E \setminus e)$ is $M_D \setminus e$ or $M_0 \setminus e$.

PROOF On the one hand, if $e \notin M_D$, removing e does not influence M_D. Thus, M_D is still a maximum matching of $(V, E \setminus e)$. On the other hand, if $e \in M_D$, while $e \notin M_0$. After removing it, M_0 can not be influenced, and M_0 is thus a maximum matching of $(V, E \setminus e)$. Besides, if $e \in M_0$ and $e \in M_D$, matching $M_D \setminus e$ and $M_0 \setminus e$ are obtained. Assume that $M_D \setminus e$ is not a maximum matching in $(V, E \setminus e)$, and the cardinality of a maximum matching must be $|M_D|$, it means that removal of e can not influence a maximum matching of D. Nevertheless, by the definition of M_0, e is contained by all maximum matchings of D in this case. Therefore, a contradiction exists, and $M_D \setminus e$ must be a maximum matching of $(V, E \setminus e)$ if $e \in M_0$ and $e \in M_D$.■

Next, according to theorem 7.1 and the minimum input theorem 2.8 of chapter 2, corollary 7.2 is concluded below to model a single edge classification:

Corollary 7.2
Given $D = (V, E)$ with M_D of definition 7.1, and M_0. Then, for any arc $e \in E$, if $e \in M_D$ and $e \in M_0$, it is a critical edge; or if $e \in M_D$ and $e \notin M_0$, it is an ordinary edge; or if $e \notin M_0$ and $e \notin M_D$, it is a redundant edge.

PROOF If $e \in E$ is in all maximum matchings of D, where $e \in M_D$ and $e \in M_0$, by theorem 7.1, its removal leads that $M_D \setminus e$ is a maximum matching of digraph $(V, E \setminus e)$. By the minimum input theorem 2.8, the minimum number of inputs to structurally control $(V, E \setminus e)$ is increased by one in number, and e is thus a critical edge. If $e \in M_D$ and $e \notin M_0$, by the minimum input theorem 2.8, removal of e does not influence M_0, by which the minimum number of inputs is still $|V| - |M_D|$, whereas the minimum set of inputs is not the same as that identified by M_D. Thus, e is an ordinary edge. If $e \notin M_D$ and $e \notin M_0$, after removing e, any maximum matching of D still exists in $(V, E \setminus e)$. By theorem 2.8, removal of e can do nothing on previously identified inputs. Thus, e is a redundant edge. ■

By corollary 7.2, the **Research question** of this chapter is thus modelled into following graph-theoretical problem:

Problem 1: Given $D = (V, E)$ with M_D of definition 7.1. Then, identify each arc that is contained by at least one maximum matching of D.

Problem 7.2.2 is eventually solved in following sections as a way to solve the research question of this chapter.

7.3 Identification of Arcs of Maximum Matchings

7.3.1 Data Structures

Given $D = (V, E)$ of definition 7.1, to identify all arcs of E that are contained by the maximum matching of D, a bipartite graph mapped by D is used, which is defined below:

Definition 7.2
Given $D = (V, E)$ and M_D of definition 7.1, let $B = (V_B, E_B)$ be a bipartite graph, V_B^+ and V_B^- be two independent vertex sets, and M_B be a maximum matching of B, where $|V_B| \leq 2|V|$, $|E_B| = |E|$, and $|M_D| = |M_B|$. Then, $V_B^+ = \{v_i^+ | 1 \leq i \leq n\}$, $V_B^- = \{v_j^- | 1 \leq j \leq n\}$ and $V_B = V_B^- \cup V_B^+$. Besides, let $\alpha : E \to E_B$ be a bijection, for each $\overrightarrow{\langle v_i, v_j \rangle} \in E$, there is $\alpha : \overrightarrow{\langle v_i, v_j \rangle} \to (v_i^+, v_j^-)$, where $(v_i^+, v_j^-) \in E_B$, $v_i^+ \in V_B^+$ and $v_j^- \in V_B^-$. Also, let M_B be mapped from M_D.

An example of the bijection is shown by figure 4.1 of chapter 4. And there can not be the edge like $(v_i^-, v_i^+) \in E_B$, because D excludes self loops. Therefore, edges of D that are contained by maximum matchings of D can be solved by finding edges of B that are contrianed by maximum matchings of B.

After that, within $B = (V_B, E_B)$, two kinds of edge sets with related to M_B are defined in definition 7.3 and 7.4, which exclude any edge of M_B. Then, through theorem 7.3, it is proved that all edges that are involved into at least one maximum matching different from M_B of B, must be only contained by these edge sets.

Definition 7.3 (Alternating-Cycle Matchig)
Given $B = (V_B, E_B)$ with M_B of definition 7.2, let $\{m_1, m_2, \ldots, m_t\}$ be a subset of M_B, and $\{e_1, e_2, \ldots, e_t\} \nsubseteq M_B$ $(2 \leq t \leq |M_B|)$ be a matching set. Then, $\{e_1, e_2, \ldots, e_t\}$ is an alternating-cycle matching related to M_B, if and only if $\{m_1, e_1, m_2, e_2, \ldots, m_t, e_t\}$ is a cycle of B.

Definition 7.4 (Alternating-Path Matching)
Given $B = (V_B, E_B)$ with M_B of definition 7.2, let $\{m_1, m_2, \ldots, m_t\}$ be a subset of M_B, and $\{e_1, e_2, \ldots, e_t\} \nsubseteq M_B$ $(1 \leq t \leq |M_B|)$ be a matching set. Then, $\{e_1, e_2, \ldots, e_t\}$ is an alternating-path matching related to M_B, if and only if $\{m_1, e_1, m_2, e_2, \ldots, m_t, e_t\}$ is a path of B.

An example of an alternating-path matching and an alternating-cycle matching are show in figure 7.1. With the alternating-path matching and alternating-cycle matching related to M_B of $B = (V_B, E_B)$ of definition 7.2, theorem 7.3 below shows how to use them related to M_B to identify edges of another different maximum matching from M_B, which is according to relationship between any two different maximum matchings of B.

Theorem 7.3
Given $B = (V_B, E_B)$ with M_B of definition 7.2. Then, for any single edge of $E_B \setminus M_B$, it is contained by at least one maximum matching of B, if and only if it belongs to an alternating-cycle matching or an alternating-path matching with respect to M_B.

Figure 7.1: An *alternating-path* matching and an *alternating-cycle matching*

The set of all blue edges of each bipartite graph above is a maximum matching, and all red edges is an alternating-path matching in $\mathbf{B_1}$, or an alternating-cycle matching in $\mathbf{B_2}$.

PROOF $\Leftarrow$: If any single edge of $E_B \setminus M_B$ belongs to an alternating-cycle matching or an alternating-path matching with respect to M_B, let $\{e_1, e_2, \ldots, e_t\}$ be such this existing alternating-cycle matching, or an alternating-path matching, which contains this single edge. By definition 7.3 and 7.4, there must be a subset of M_B, and edges of this set is adjacent to the same number of edges of $\{e_1, e_2, \ldots, e_t\}$. Let $\{m_1, m_2, \ldots, m_t\} \subseteq M_B$ be such a set. Then, replacing $\{m_1, m_2, \ldots, m_t\} \subseteq M_B$ with $\{e_1, e_2, \ldots, e_t\}$ can generate a maximum matching of B, which is $M_B \setminus \{m_1, m_2, \ldots, m_t\} \cup \{e_1, e_2, \ldots, e_t\}$. Therefore, $\{e_1, e_2, \ldots, e_t\}$ is contained by at least one different maximum matching from M_B, let along that single edge.

$\Rightarrow$: If a single edge of $E_B \setminus M_B$ is contained by a maximum matching of B, let M_{B_0} be such a maximum matching, and $M_{B_0} \neq M_B$. Then, because $M_B \oplus M_{B_0}$ must contain vertex-disjoint paths or cycles, which alternatively include the same number of edges of M_B and M_{B_0}. Therefore, let $\{m_1, e_1, m_2, e_2, \ldots, m_t, e_t\}$ be either a single path or a cycle of $M_B \oplus M_{B_0}$, where $1 \leq t, \leq |M_B|$, $\{m_1, m_2, \ldots, m_t\} \subseteq M_B$ and $\{e_1, e_2, \ldots, e_t\} \subseteq M_{B_0}$. By definition 7.3, and definition 7.4, $\{e_1, e_2, \ldots, e_t\}$ is either an alternating-path matching or an alternating-cycle matching with respect to M_B. And the single edge of $E_B \setminus M_B$ contained by M_{B_0} belongs to an alternating-path matching or an alternating-cycle matching with respect to M_B. ■

As a result, for any edge of one of these identified edge sets, noted by $(v_i^+, v_j^-) \in E_B$, by the bijection of definition 7.2, arc $\overrightarrow{\langle v_i, v_j \rangle}$ of $D = (V, E)$ of definition 7.1 can be confirmed as an arc that is contained by at least one maximum matching of D. Procedure of identification is done in next section 7.3.2.

7.3.2 Execution

Given $B = (V_B, E_B)$ with M_B of definition 7.2, to identify all alternating-cycle matchings and alternating-path matchings related to M_B with lower efficiency, it is also indispensable to understand the distribution of them within B, which is clarified by theorem 7.4 and theorem 7.5.

Theorem 7.4
Given $B = (V_B, E_B)$ with M_B of definition 7.2, let $v_i^+ \in V_B^+$, $v_j^- \in V_B^-$ be not incident to any edges of M_B. Then, related to M_B, any two distinct alternating-path matchings incident to v_i^+ and v_j^- respectively, must be vertex-disjoint.

Figure 7.2:

PROOF Assume that there exists a node shared by two alternating-path matchings incident to v_i^+ and v_j^-, respectively. And this shared node is involved into a path incident to v_i^+ and alternatively involving edges of M_B and $E_B \setminus M_B$. Simultaneously, this shared node is also involved into another path incident to v_j^- and alternatively involving edges of M_B and $E_B \setminus M_B$. As a result, these two alternating paths related to M_B construct an augmenting path of definition 2.12 of chapter 2 related to M_B, whose existence in B contradicts with the maximality of M_B. For example, in figure 7.2 **a**, blue lines are a matching, $\{(v_2^+, v_4^-), (v_3^+, v_1^-)\}$ and $\{(v_4^+, v_3^-)\}$ are two alternating-path matchings related to this blue-line matching. Then, $\{(v_4^+, v_3^-), (v_3^-, v_2^+), (v_2^+, v_4^-), (v_4^-, v_3^+), (v_3^+, v_1^-)\}$ is an augmenting path. Thus, any two alternating-path matchings incident to v_i^+ and v_j^- respectively, must be vertex-disjoint. ■

After investigating distribution among alternating-path matchings, theorem 7.5 illustrates distribution among both alternating-cycle matchings and alternating-path matchings with respect to M_B.

Theorem 7.5
Given $B = (V_B, E_B)$ with M_B of definition 7.2, let $v_i^+ \in V_B^+$, $v_j^- \in V_B^-$ be not incident to any edges of M_B. Then, related to M_B, any two distinct alternating-path matchings incident to v_i^+ and v_j^- respectively, can not be adjacent to a same alternating-cycle matching.

PROOF By definition 7.3 and 7.4, any alternating-cycle matching related to M_B can be adjacent to a single alternating-path matching, and any vertex incident to an edge of an alternating-cycle matching is also involved into a cycle that alternatively includes edges of M_B and $E_B \setminus M_B$. Furthermore, any shared vertex by an alternating-cycle matching and an alternating-path matching must be involved into an alternating path related to M_B. Because of this, such an alternating-cycle matching can not be adjacent to two distinct alternating-path matching incident to v_i^+ and v_j^- respectively. Otherwise, an augmenting path related to M_B would exist. For example, in figure 7.2 **b**, all blue lines are a matching, with respect to it, $\{(v_2^+, v_4^-), (v_3^+, v_2^-), (v_1^+, v_3^-)\}$ is an alternating-cycle matching, while $\{(v_4^+, v_3^-)\}$ and $\{(v_3^+, v_1^-)\}$ are two alternating-path matchings. Then, $\{v_4^+, v_3^-), (v_3^-, v_2^+), (v_2^+, v_4^-), (v_4^-, v_3^+), (v_3^+, v_1^-)\}$ is an augmenting path. Hence, this theorem holds correctness.■

After this, to identify all alternating-path and alternating-cycle matchings with respect to M_B of $B = (V_B, E_B)$ of definition 7.2, a digraph is used, which is noted

by $D_0 = (V_0, E_0)$ and produced by algorithm 7.1 below, where any alternating-path and alternating-cycle matching related to M_B are corresponding to a directed path and a cycle of D_0. In algorithm 7.1, given $B = (V_B, E_B)$ with M_B, let V_0 and E_0 be an initially empty vertex and arc set, respectively. Also, let (v_i^+, v_j^-) be any single edge of E_B, and (v_p^+, v_q^-) be any single edge of M_B. Besides, S_1 represents an initially empty set of vertices, and let $\overrightarrow{\langle v_p, v_q \rangle}$ and $\overrightarrow{\langle v_i, v_j \rangle}$ be two arcs.

Algorithm 7.1: Produce a Digraph $D_0 = (V_0, E_0)$.

Input : $B = (V_B, E_B)$ with M_B of definition 7.2, E_0, V_0, S_1.
Output: Digraph $D_0 = (V_0, E_0)$.

1 **while** $E_B \neq \emptyset$ ***and*** $(v_i^+, v_j^-) \in E_B$ **do**
2 $\quad E_B^{'} = E_B \setminus (v_i^+, v_j^-)$;
3 $\quad$ **if** $(v_i^+, v_j^-) = (v_p^+, v_q^-)$ **then**
4 $\quad\quad$ Colour $\overrightarrow{\langle v_i, v_j \rangle}$ into red;
5 $\quad E_0^{'} = E_0 \cup \overrightarrow{\langle v_i, v_j \rangle}$;
6 $\quad V_0^{'} = V_0 \cup \{v_i, v_j\}$;
7 **while** $|E_0| \geq |E_B \setminus M_B|$ ***and*** $\exists \overrightarrow{\langle v_p, v_q \rangle}$ *coloured* **do**
8 $\quad E_0^{'} = E_0 \setminus \overrightarrow{\langle v_p, v_q \rangle}$;
9 $\quad$ Replace $\{v_p, v_q\}$ with v_{pq} in V_0;
10 $\quad$ Replace arcs whose heads are v_q with arcs whose heads are v_{pq} in E_0;
11 $\quad$ Replace arcs whose tails are v_p with arcs whose tails are v_{pq} in E_0;
12 Identify every node of V_0 that is not obtained by line 9; **and** Add it into S_1;
13 **return** $D_0 = (V_0, E_0)$; S_1;

PROOF Initially, in the first **while** loop, each edge of E_B is directed from vertices of V_B^+ to vertices of V_B^-, and those arcs are added into E_0, where arcs mapped by edges of M_B are marked by colouring. Also, nodes incident to those arcs are also added into V_0. Then, in the second **while** loop, each arc $\overrightarrow{\langle v_p, v_q \rangle} \in E_0$ that is produced by an edge $(v_p^+, v_q^-) \in M_B$ is identified and removed from E_0. Also, each vertex pair $\{v_p, v_q\}$ is together replaced by a single vertex v_{pq}, so that v_{pq} is the tail of the arc whose tail was v_p, and v_{pq} is also the head of the arc whose head was v_q. Therefore, each arc of E_0 after running the second **while** loop can construct directed paths or cycles, each of which is thus corresponding to one exact alternating-path or cycle matching with respect to M_B. Because each considered edge of E_B and M_B is removed, the first and the second **while** loop must terminate at a moment. An example of this algorithm is shown in figure 7.3. ∎

Corollary 7.6 (Time complexity of algorithm 7.1)
Given $B = (V_B, E_B)$ with M_B of definition 7.2, the worst-case execution time of running algorithm 7.1 is $O(|V_B| + |E_B|)$.

PROOF Because each considered edge of E_B and M_B is removed in line 2 and 8, respectively, the two loops together run $O(|E_B|)$ steps. Then, identifying each node of V_0 at line 12 costs $O(|V_0|)$. Besides, focusing on line 9, because $2|V_0| \leq |V_B|$,

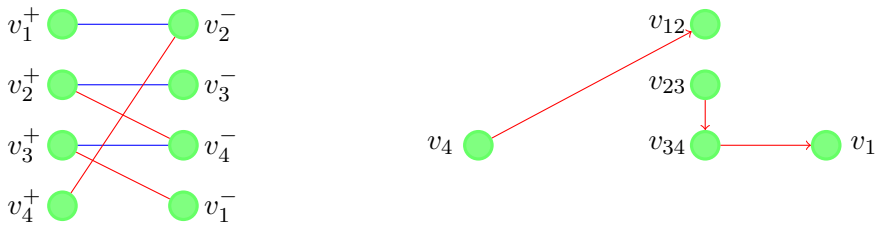

Figure 7.3:

The bipartite graph contains a maximum matching of all blue edges, which is on the left and is mapped into a digraph on the right based on algorithm 7.1.

worst-case execution time of this algorithm is $O(|V_B| + |E_B|)$, which excludes time cost by deriving inputs. ■

Additionally, in D_0, each alternating-path matching related to M_B of B is mapped into a directed path starting from or ending at vertices of S_1, while each alternating-cycle matching is mapped into a directed cycle only involving vertices like v_{pq}, where $p \neq q$, and any single vertex like v_i and v_j of V_0 can not be connected, otherwise, there would be an augmenting path related to M_B. By theorem 7.4 and 7.5, those directed paths corresponding to alternating-path matchings incident to $v_i^+ \notin M_B$ and $v_j^- \notin M_B$ must be disjoint as well. Also, there could be multiple paths of D_0 sharing a same terminal vertex, which correspond to alternating-path matchings incident to a same vertex out of M_B.

Next, algorithm 7.2 finds all those paths and cycles corresponding to alternating-path and cycle matchings within $D_0 = (V_0, E_0)$ returned by algorithm 7.1. Clearly, cycles of D_0 are actually contained by strongly connected components of D_0. In this algorithm, for S_1 returned by algorithm 7.1, let any single vertex of S_1 be v_i. Also, let S_2 be an initially empty arc set.

Algorithm 7.2: Identify arcs of directed paths and cycles.

Input : $D_0 = (V_0, E_0)$ and S_1 returned by algorithm 7.1, S_2.
Output: Arcs of directed paths and cycles.

```
while S_1 ≠ ∅ and v_i ∈ S_1 do
    S_1' = S_1 \ v_i;
    Identify all arcs of E_0 through paths that start from or end at v_i without
      repetition;
    Add identified arcs into S_2;
    Remove all identified arcs from E_0;
Identify arcs of strongly connected components of remaining D_0 by Tarjan's
  algorithm of [111];
Add identified arcs into S_2;
return S_2;
```

PROOF By running algorithm 7.1, each alternating-path matching related to M_B of $B = (V_B, E_B)$ of definition 7.2 corresponds to a directed path starting from or ending at a node of S_1, while any alternating-cycle matching related to M_B is used to

construct a directed cycle of D_0, which only involves nodes of $V_0 \setminus S_1$. Therefore, in line 3, starting from each $v_i \in S_1$, all arcs of paths starting from or ending at v_i can be identified and added into S_2. Particularly, this identification process can not be based on breadth-first or depth-first search, because some arcs might be not traversed, otherwise. Instead, as long as an arc is involved into a path whose terminal is v_i, it should be traversed once and added into S_2, whenever its head and tail have been visited previously or not. Because of line 2, this **while** loop terminates at the moment when $S_1 = \emptyset$. At the same time, each traversed edge is removed from E_0. After this, by the strongly-component identification algorithm, all cycles that are not joint with paths can be found and added into S_2. Therefore, each cycle and path can be traversed by this algorithm, and this algorithm is correct. ■

Corollary 7.7 (Time complexity of algorithm 7.2)
Given $D_0 = (V_0, E_0)$ returned by algorithm 7.1, the worst-case execution time of running algorithm 7.2 is $O(|V_B| + |E_B|)$.

PROOF Due to line 5, each arc of D_0 is traversed once only, and the **while** loop costs $O(|E_0|)$ steps in the worst case. Besides, running Tarjan's strongly connected-component identification algorithm can identify all arcs of cycles of D_0. For the worst-case execution time, except for obtaining inputs, it is $O(|V_0| + |E_0|)$, which is cost by line 3 and line 6. Because $|E_0| < |E_B|$ and $|V_0| < |V_B|$ according to algorithm 7.1, time complexity of this algorithm is also represented by $O(|V_B| + |E_B|)$. ■

With S_2, the following algorithm 7.3 identifies each arc of $D = (V, E)$ of definition 7.1, which is contained by a maximum matching, so that categories of all arcs of D can be confirmed by theorem 7.2. Here, any single arc of S_2 is noted by e_0. In notation of e_0, according to algorithm 7.1, it could be represented by one of four cases: $\overrightarrow{\langle v_i, v_{pq}\rangle}$, $\overrightarrow{\langle v_{pq}, v_i\rangle}$, $\overrightarrow{\langle v_{ij}, v_{pq}\rangle}$, $\overrightarrow{\langle v_{pq}, v_{ij}\rangle}$ where v_i, v_{pq}, v_{ij} are vertices of V_0.

PROOF According to algorithm 7.2, any arc of S_2 is involved into either a directed cycle or a path, which corresponds to an alternating-path or an alternating-cycle matching related to M_B of $B = (V_B, E_B)$ of definition 7.2 by algorithm 7.1. Also, by the bijection of definition 7.2 and theorem 7.3, all those alternating-path and cycle matchings are mapped by arcs of maximum matchings that are different from M_D. Therefore, the remaining M_D of line 12 only contains all arcs that are shared by each maximum matching of D. By contrast, the finally returned E contains arcs that are out of all maximum matchings of D. Also, because each edge of S_2 is removed, the **while** loop would terminate when $S_2 = \emptyset$. Above all, this algorithm is correct. ■

Corollary 7.8 (Time complexity of algorithm 7.3)
Given $D = (V, E)$ with M_D of definition 7.1, and S_2 returned by algorithm 7.2, the worst-case execution time of running algorithm 7.3 is $O(|E|)$.

PROOF For the time complexity, since each arc of S_2 is removed, and identifying an arc of D by any given e_0 costs $O(1)$, running the **while** loop thus costs $O(|S_2|)$ steps. Due to $|S_2| < E_0$, $|E_0| < |E_B|$ by algorithm 7.1, and $|E_B| = |E|$ by definition 7.2. Time complexity of this algorithm can also be presented by $O(|E|)$, which excludes deriving the inputs of this algorithm. ■

Algorithm 7.3: Identify arcs contained by maximum matchings of D.

Input : $D = (V, E)$ with M_D of definition 7.1, S_2 returned by algorithm 7.2.
Output: Arcs of maximum matchings.

while $S_2 \neq \emptyset$ ***and*** $e_0 \in S_2$ **do**
  $S_2^{'} = S_2 \setminus e_0$;
  if $e_0 = \overrightarrow{\langle v_i, v_{pq} \rangle}$ ***or*** $e_0 = \overrightarrow{\langle v_{pq}, v_i \rangle}$ **then**
    $M_D^{'} = M_D \setminus \overrightarrow{\langle v_p, v_q \rangle}$;
    $E^{'} = E \setminus \{\overrightarrow{\langle v_i, v_q \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$ **or** $E^{'} = E \setminus \{\overrightarrow{\langle v_p, v_i \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$;
    return $\{\overrightarrow{\langle v_i, v_q \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$ **or**$\{\overrightarrow{\langle v_p, v_q \rangle}, \overrightarrow{\langle v_p, v_i \rangle}\}$;
    else if $e_0 = \overrightarrow{\langle v_{ij}, v_{pq} \rangle}$ ***or*** $e_0 = \overrightarrow{\langle v_{pq}, v_{ij} \rangle}$ **then**
      $M_D^{'} = M_D \setminus \{\overrightarrow{\langle v_p, v_q \rangle}, \overrightarrow{\langle v_i, v_j \rangle}\}$;
      $E^{'} = E \setminus \{\overrightarrow{\langle v_i, v_j \rangle}, \overrightarrow{\langle v_i, v_q \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$ **or**
       $E^{'} = E \setminus \{\overrightarrow{\langle v_i, v_j \rangle}, \overrightarrow{\langle v_p, v_j \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$;
      return $\{\overrightarrow{\langle v_i, v_j \rangle}, \overrightarrow{\langle v_i, v_q \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$ **or** $\{\overrightarrow{\langle v_i, v_j \rangle}, \overrightarrow{\langle v_p, v_j \rangle}, \overrightarrow{\langle v_p, v_q \rangle}\}$;
$E^{'} = E \setminus M_D$;
return E, M_D;

After that, by corollary 7.2, those returned arcs via each e_0 are classified into ordinary category, and arcs of returned M_D are classified into critical category, while arcs of returned E are classified into redundant category.

7.3.3 Time Complexity Analysis

Given $D = (V, E)$ and M_D of definition 7.1, the entire process of identifying arcs contained by maximum matchings of D is composed following procedures:

1. Derive $B = (V_B, E_B)$ with M_B of definition 7.2;

2. Run algorithm 7.1, 7.2 and 7.3, in sequence.

Except for deriving $D = (V, E)$ and M_D, to solve problem 7.2.2, the worst-case execution time of entire process is the sum of time complexity of each procedure above. Obviously, deriving $B = (V_B, E_B)$ and M_B costs $O(|E|)$ based on the bijection of definition 7.2. Then, running those three algorithms costs $O(|V_B| + |E_B)$ together. Meanwhile, because $|E_B| = |E|$ and $|V_B| \leq 2|V|$, the worst-case execution time is $O(|V|+|E|)$. When identifying M_D of D is considered, time complexity becomes $O(|V| + |E| + \sqrt{|V|} \cdot |E|)$.

7.3.4 Comparison

By contrast, with the same assumptions, obviously, performance of procedure shown above is more efficient than using the algorithm of [97] to identify all arcs of maximum matchings of D. Additionally, since all those identified arcs by the process shown above, are also called the maximally-matchable edges of D, compared with

algorithms of [112], each arc included by at least one maximum matching of D is always identified. In summary, all arcs of $D = (V, E)$ can be efficiently and accurately classified into critical, redundant and ordinary categories, respectively.

7.4 Summary

To effectively identify vulnerable edges to the removal and in order to effectively maintain network structural controllability with a minimum set of inputs, this chapter efficiently classifies arcs of a minimum-input structurally controllable CT-LTI network into critical, redundant and ordinary categories. And the solution is to identify all edges of maximum matchings of the input network, where one bipartite graph and a directed graph are used and it is mapped by the input digraph. As a result, this solution can be executed more accurately and efficiently than existing related works shown in section 3.4.1 of chapter 3. After edge-based analysis, in following chapters, vertex-based security analysis for structural controllability would be implemented.

Chapter 8

Driver-Node based Analysis for Structural Controllability

8.1 Overview

After implementing the edge-based analysis to maintain structural controllability with a minimum set of inputs, the vertex-based analysis for the same purpose would be considered from this chapter onwards. In this chapter, the driver-node based analysis is executed. As discussed in section 2.6 of chapter 2, to structurally control a CT-LTI network, such as $G(\mathbf{A}) = (V_1, E_1)$ of definition 2.4, driver nodes are indispensable network vertices, which are directly forced by external inputs. Due to such importance, driver nodes would be undoubtedly targeted by malicious attackers. For example, attackers can affect found driver nodes by themselves to hijack current structural control into the targeted network [34]. Besides, attackers directly damage found driver nodes to disrupt structural control. In particular, recently, Jia $et\ al.$ [62] raised an effective method to identify each single network vertex that is included by all minimum sets of driver nodes in polynomial time. In detail, given a CT-LTI dynamical network with n vertices and m edges in number, identifying all nodes of each minimum set of driver nodes costs $O(m{\cdot}n)$ steps in the worst case. Clearly, for attackers, compared with identifying only one set of driver nodes in each time, this method is more efficient and convenient to hijack and damage current structural controllability. Additionally, it is also essential to identify vulnerable driver nodes to the removal. This is because previous works listed in section 3.4.2 of chapter 3 did not further investigate the harmfulness caused by removing a single driver node.

Therefore, this chapter solves the research question 5. In detail, with assumptions of section 1.3.2 of chapter 1, it is also assumed there is a CT-LTI dynamical digraph as an input network. This chapter firstly identifies every vertex of all minimum sets of driver nodes,which is executed more efficiently than [62]. Then, in order to explicitly identify vulnerable driver nodes to the removal, this chapter classifies each single driver node based on impacts of its removal on structural control into the residual network. Additionally, the minimum set of driver nodes is obtained by the maximum-matching based method shown in section 2.4.1 of chapter 2, in assumption.

For the solution, this chapter identifies all vertices of minimum sets of driver nodes by finding each single vertex, which is an unmatched node related to a maximum matching of the input network. According to the minium input theorem [73], this is because any minimum set of driver nodes obtained by the maximum-matching based method is also a set of unmatched nodes. After this, for the same reason, each drive node is classified by the impact of its removal on the minimum

set of unmatched nodes with respect to a maximum matching of the residual network. It is concluded that removing a driver node either reduces the original cardinality of the minimum set of driver nodes by one, or not change. And those left driver nodes can still be used to structurally control the residual network. For the whole operations about identification and classification of all existing single driver nodes, they are executed within a bipartite graph. This bipartite graph is mapped by the input network. As a result of running those operations, the worst-case execution time is linear to identify each single node that can be a driver node. This cost is more efficient than the result of running the method of [62].

For the contribution of this chapter, given a CT-LTI dynamical digraph, vertices of all minimum sets of driver nodes are identified in $O(n \cdot \sqrt{m})$ steps at most. To further understand the harmfulness caused by a single driver-node removal, those identified nodes are also efficiently classified by impacts of any single driver-node removal on structurally controlling the residual network.

Remaining chapter is structured as follows: section 8.2 specifies research question and gives preliminaries; section 8.3 shows related solution; and the last section 8.4 summarizes this chapter.

8.2 Preliminaries & Problem Formulation

8.2.1 Research Question

Firstly, the input network of this chapter is defined below:

Definition 8.1 (Input Network of chapter 8)
Let $D = (V, E)$ be a large and finite digraph, excluding self loops, parallel arcs and isolated nodes, where vertex set $V \neq \emptyset$, $V = \{v_i | 1 \leq i \leq n\}(n > 2)$, and arc set $E \neq \emptyset$, $|E| > 2$, $E = \{\overrightarrow{\langle v_i, v_j \rangle} | i \neq j, v_i, v_j \in V\}$.

Then, assume that $D = (V, E)$ can be represented by following CT-LTI dynamics:

$$\dot{x}(t) = \mathbf{A}x(t) \tag{8.1}$$

where, $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $D = (V, E)$ is assumed to be mapped into the matrix $\mathbf{A}$. Also, each arc noted by $\overrightarrow{\langle v_i, v_j \rangle} \in E$, only corresponds to one non-zero entry noted by $a_{ji} \in \mathbf{A}$. Then, with $D = (V, E)$ of definition 8.1, research question of this chapter is formally defined:

Research Question: Given $D = (V, E)$ of definition 8.1, efficiently identify each single vertex that could be included by a minimum set of driver nodes, and also classify those identified nodes based on how any single driver node maintains the currently used minimum set of driver nodes.

8.2.2 Modelling

By the maximum-matching based method of section 2.4.1 of chapter 2, or the minimum input theorem 2.8 of section 2.6, because each minimum set of driver nodes could be a set of unmatched nodes related to a maximum matching of a given

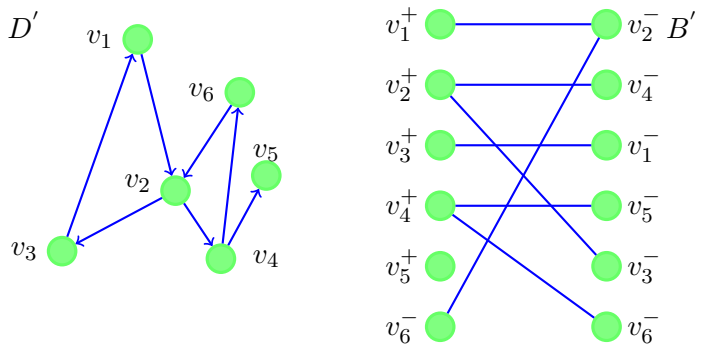

Figure 8.1:

A digraph D' is mapped into a bipartite graph B' by definition 8.2.

digraph with CT-LTI dynamics. Therefore, **Research Question** of this chapter is modelled by following problem:

Problem 1: Within $D = (V, E)$ of definition 8.1, identify each single vertex that is included by a set of unmatched nodes related to a maximum matching of D, and also classify them based on how any one of them maintains the currently identified unmatched nodes with respect to a maximum matching of D.

Furthermore, problem 8.2.2 is modelled by **Problem 2** below with a bipartite graph of definition 8.2 according to lemma 8.1.

Definition 8.2
Given $D = (V, E)$ of definition 8.1, let $B = (V_B, E_B)$ be a bipartite graph, V_B^+ and V_B^- be two disjoint and independent sets of V_B, where $|E_B| = |E|$, $|V_B| = 2|V|$, $|V_B^-| = |V_B^+|$, and $V_B = \{\{v_i^-, v_i^+\} | v_i^- \in V_B^-, v_i^+ \in V_B^+\}$. Besides, let $\beta : V \to V_B$, and $\gamma : E \to E_B$ be two different bijections. For any $v_i \in V$, $\beta : v_i \to \{v_i^+, v_i^-\}$, where $v_i^- \in V_B^-, v_i^+ \in V_B^+$; for any $\overrightarrow{\langle v_i, v_j \rangle} \in E$, $\gamma : \overrightarrow{\langle v_i, v_j \rangle} \to (v_i^+, v_j^-)$, where $(v_i^+, v_j^-) \in E_B$, $v_i^- \in V_B^-$ and $v_j^+ \in V_B^+$.

An example of bijections above is shown by figure 8.1. And there can not be the edge like $(v_i^-, v_i^+) \in E_B$, because D excludes self loops.

Then, lemma 8.1 uses $B = (V_B, E_B)$ of definition 8.2 to identify unmatched nodes with respect to maximum matchings of $D = (V, E)$ of definition 8.1:

Lemma 8.1
Given $D = (V, E)$ and $B = (V_B, E_B)$, let v_i be a single node of V, and $\{v_i^-, v_i^+\}$ be a pair of nodes of V_B and mapped by v_i. Also, let M_D be a maximum matching of D, and M_B be a maximum matching of B, where M_D is assumed to be mapped into M_B by γ of definition 8.2. Then, $v_i \in V$ is an unmatched node with respect to M_D, if and only if v_i^- is an unmatched node in V_B^- with respect to M_B.

PROOF $\Rightarrow$: If $v_i \in V$ in D is an unmatched node with respect to M_D, and M_B is mapped by M_D. By definition 8.2, $\{v_i^-, v_i^+\} \subseteq V_B$ is mapped from v_i, and $v_i^- \in V_B^-$ can not be incident to any edge of M_B. Thus, v_i^- is unmatched related to M_B in V_B^-.

$\Leftarrow$: If v_i^- is an unmatched node in V_B^- related to M_B. By definition 8.2, v_i^- can be only incident to edges out of M_B. Let $(v_j^+, v_i^-) \notin M_B$ be such an edge. Then, there

is $\gamma^{-1} : (v_j^+, v_i^-) \rightarrow \overrightarrow{\langle v_j, v_i \rangle}$, and $\overrightarrow{\langle v_j, v_i \rangle} \notin M_D$ as a result. Thus, v_i is unmatched related to M_D in D. ■

By lemma 8.1, following problem is defined below to model problem 8.2.2:

Problem 2: Given $B = (V_B, E_B)$ of definition 8.2, find each vertex of V_B^-, which is an unmatched node related to a maximum matching of B. Besides, let $\{v_i^-, v_i^+\}$ be a pair of nodes of V_B, where v_i^- is an unmatched node related to a maximum matching of B, and then classify those found nodes based on how any $\{v_i^-, v_i^+\}$ maintains the current unmatched nodes of V_B^- with respect to a maximum matching of B.

Problem 2 is eventually solved in following as a way to solve the problem 8.2.2 and research question of this chapter, further.

8.3 Solution

8.3.1 Unmatched-node Identification

This section identifies all nodes of V_B^- that are unmatched nodes with respect to maximum matchings of bipartite graph $B = (V_B, E_B)$ of definition 8.2. To do this, lemma 8.2 identifies a vertex of V_B^-, which is a matched node with respect to a given maximum matching, but it is also an unmatched node related to another different maximum matching:

Lemma 8.2
In $B = (V_B, E_B)$ of definition 8.3, let M_i be a maximum matching of B. With respect to M_i, let $v_i^- \in V_B^-$ be a matched node, and $v_j^- \in V_B^-$ be an unmatched node. Then, with respect to a maximum matching different from M_i, v_j^- is matched, while v_i^- becomes unmatched, if and only if v_i^- and v_j^- are connected by an existing path of B, which alternatively involves edges of M_i and $E_B \setminus M_i$.

PROOF Let M_j be a different maximum matching from M_i and involving an edge incident to v_j^-.

$\Rightarrow$: Related to M_j, if v_i^- is an unmatched node, while v_j^- is matched. Because $M_i \oplus M_j$ contains vertex-disjoint paths, or cycles that alternatively involve the same number of edges of M_i and M_j, v_i^- and v_j^- can be only involved into a path. Otherwise, either v_j^- is matched related to M_i, or v_i^- is matched related to M_j, which is a contradiction. Hence, v_i^- and v_j^- are connected by a path alternatively involving edges of M_i and M_j in B.

$\Leftarrow$: If v_i^- and v_j^- are connected by a path alternatively involving edges of M_i and $E_B \setminus M_i$. Let P be this path, whose two ending nodes must be v_i^- and v_j^-, respectively. Then, P must contain the same number of edges of M_i and $E \setminus M_i$. Otherwise, v_i^- and v_j^- can not be connected. Thus, a different maximum matching of B can be obtained by: $M_i \oplus P$. Meanwhile, with respect to $M_i \oplus P$, v_i^- is an unmatched node and v_j^- is matched. ■

According to lemma 8.2, let $v_i^+ \in V_B^+$ be a matched node related to a maximum matching, noted by M_i, we can also know how v_i^+ is an unmatched node related to

a different maximum matching. Then, in remaining parts of this chapter, any single vertex like v_i^- of lemma 8.1 is called the *extra-unmatched node*, which is formally defined below:

Definition 8.3 (Extra-unmatched Node)
Given $B = (V_B, E_B)$ of definition 8.2, let M_i, M_j be two different maximum matchings of B. Then, $v_i^- \in V_B^-$ is an extra-unmatched node of V_B^- via M_i, if and only if $v_i^- \in V_B^-$ is a matched node with respect to M_i, while it is also an unmatched node with respect to M_j.

Remark 3 By definition 8.3, with respect to M_i, a matched node of V_B^+ that is unmatched node with respect to a different maximum matching from M_i, is also called an *extra-unmatched node of V_B^+ via M_i* accordingly. □

However, based on the lemma 8.2, an inevitable question emerges:

"*Given $B = (V_B, E_B)$ of definition 8.2, let M_i be a maximum matching of B, when the number of all* extra-unmatched nodes of V_B^- via M_i *is less than $|M_i|$, for any node of V_B^- that is not an* extra-unmatched node of V_B^- via M_i*, whether it is an extra-unmatched node of V_B^- via a different maximum matching from M_i, or not?*"

Obviously, without clarifying this issue, it is still unknown if all extra-unmatched nodes of V_B^- via M_i and all unmatched nodes of V_B^- with respect to M_i, are all unmatched nodes related to maximum matchings of B. Hence, theorem 8.3 and corollary 8.4 below are deduced to make such a clarification:

Theorem 8.3
Given $B = (V_B, E_B)$ of definition 8.2, let M_i, M_j be any two different maximum matchings of B, and let v_i^- be a matched node related to both M_i and M_j. Besides, assume that v_i^- is not an extra-unmatched node of V_B^- via M_i*. Then, v_i^- is also not an extra-unmatched node of V_B^- via M_j.*

PROOF Assume that v_i^- is an extra-unmatched node of V_B^- via M_j, and let P_j be an existing path by lemma 8.2, which alternatively involves edges of M_j and $E_B \setminus M_j$ and connects v_i^- with an unmatched node of V_B^- related to M_j. Accordingly, by proving that P_j can not exist, v_i^- can therefore not be an extra-unmatched node of V_B^- via M_j, and correctness of this theorem is proved eventually.

Further, let $h(P_j)$ be a subset of P_j, where $|P_j| = 2|h(P_j)|$ and $h(P_j) \subseteq M_j$. Then, by lemma 8.2, an edge of $h(P_j)$ is therefore incident to v_i^-. Also, since there are no edges of M_j that are disjoint with edges of M_i, otherwise, maximality of M_i is contradicted. As results, some edges of M_j either construct disjoint paths or cycles with the same number of edges of M_i, while other edges of M_j can be shared with by M_i. Based on these relationships among edges of M_i and M_j, we next investigate whether $h(P_j)$ exists or not in following representative cases.

1. If $h(P_j) \subseteq \{M_i \cap M_j\}$. Then, P_j that contains an edge incident to v_i^-, can alternatively contain edges of M_i and $E_B \setminus M_i$, and v_i^- is thus an extra-unmatched node of V_B^- via M_i by lemma 8.2, which contradicts to that v_i^- is not an extra-unmatched node of V_B^- via M_i. Therefore, this case is invalid.

2. If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$ and $h(P_j)$ only constructs a path with the same number of edges of M_i. Also, this path can not connect an unmatched node of V_B^- with respect to M_j. Otherwise, v_i^- is an unmatched node of V_B^- via M_i.

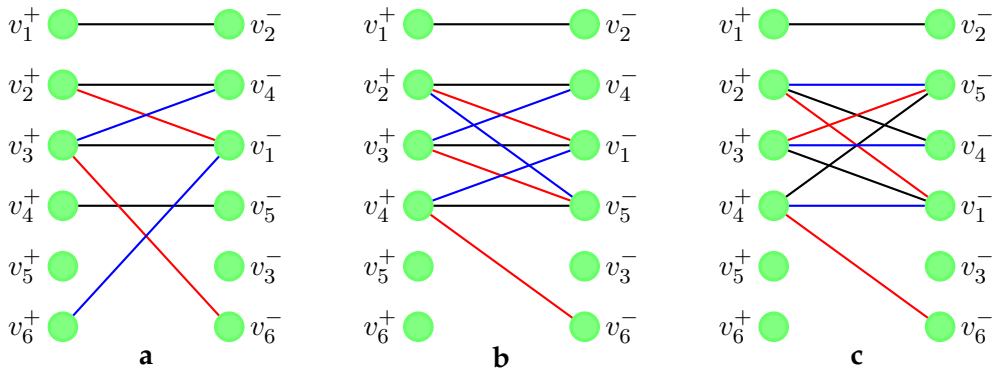

Figure 8.2:

Let $e_i = (v_k^+, v_i^-)$ be an edge of $h(P_j)$ and $e_i \notin M_i$. Then, this path that alternatively involves edges of M_i and M_j, can only connect v_k^+ with an unmatched node of V_B^+ related to M_j. We denote this path as P_i. Also, since $h(P_j) \subseteq P_j$, vertices of P_j are shared with P_i, P_j and P_i can construct an augmenting path with respect to M_j, so that the cardinality of M_j is augmented by corollary 2.3 of chapter 2, which contradicts the maximality of M_j. Thus, P_i can not exist and this case is invalid.

For example, in figure 8.2 **a**, $M_j = \{(v_1^+, v_2^-), (v_2^+, v_4^-), (v_3^+, v_1^-), (v_4^+, v_5^-)\}$, and $h(P_j) = \{(v_2^+, v_4^-), (v_3^+, v_1^-)\}$, and $P_j = h(P_j) \cup \{(v_2^+, v_1^-), (v_3^+, v_6^-)\}$. Also, $P_i = h(P_j) \cup \{(v_3^+, v_4^-), (v_6^+, v_1^-)\}$. Obviously, an augmenting path related to this maximum matching is $\{(v_6^+, v_1^-), (v_1^-, v_3^+), (v_3^+, v_6^-)\}$.

3. If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$ and $h(P_j)$ only constructs a cycle with the same number of edges of M_i. Because all vertices of this cycle are connected, nodes of this cycle and V_B^- are shared by both P_j and M_i. Thus, there is a path alternatively involving edges of $E_B \setminus M_i$ and M_i, so that v_i^- is an extra-unmatched node of V_B^- via M_i, which is a contradiction, and this case is invalid.

   For example, in figure 8.2 **b**, $M_j = \{(v_1^+, v_2^-), (v_2^+, v_4^-), (v_3^+, v_1^-), (v_4^+, v_5^-)\}$, and $h(P_j) = \{(v_2^+, v_4^-), (v_3^+, v_1^-), (v_4^+, v_5^-)\}$, and $P_j = h(P_j) \cup \{(v_2^+, v_1^-), (v_3^+, v_5^-), (v_4^+, v_6^-)\}$. Also, the cycle is $h(P_j) \cup \{(v_3^+, v_4^-), (v_2^+, v_5^-), (v_4^+, v_1^-)\}$. Then, in figure 8.2 **c**, which is completely same as the graph of **b**, $M_i = \{(v_1^+, v_2^-), (v_2^+, v_5^-), (v_3^+, v_4^-), (v_4^+, v_1^-)\}$. Obviously, due to existence of that cycle, v_5^-, v_4^-, v_1^- are extra-unmatched nodes of V_B^- via M_i, which are contradictions with $h(P_j)$.

4. If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$ and $h(P_j)$ constructs both a path and a cycle with edges of M_i. Because v_i^- can be incident to an edge of either this cycle or path, and such a cycle or path is the cycle or path of previous invalid case two or three. And such path and cycle are also vertex disjoint. As a result, referring to the contradiction occurring in cases two or three, this case is also invalid.

5. If $h(P_j)$ contains edges of both $M_i \cap M_j$ and $M_j \setminus \{M_i \cap M_j\}$. For the same reason of impossibility of case four above, or the contradiction of case one, or both of them, this case is also impossible.

Now, we prove that those five invalid cases cover all possible cases that v_i^- is an extra-unmatched node of V_B^- via M_j. Because $h(P_j) \subseteq \{M_i \cap M_j\}$ or $h(P_j) \subseteq \{M_i \setminus \{M_i \cap M_j\}\}$, or $h(P_j)$ contains edges of these two independent sets. Also, because $h(P_j)$ constructs a path or cycle or both path and cycle with the same number of edges of M_i. Therefore, in combination of those conditions, there might be nine cases in total, while those five cases above can be further extended to them as follows with other specific conditions:

1. If $h(P_j) \subseteq \{M_i \cap M_j\}$, and

   a) $h(P_j)$ constructs a path with edges of M_i; or

   b) $h(P_j)$ constructs a cycle with edges of M_i; or

   c) $h(P_j)$ constructs a path and a cycle with edges of M_i.

2. If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$, and

   a) $h(P_j)$ only constructs a path with edges of M_i; or

   b) $h(P_j)$ only constructs a cycle with edges of M_i; or

   c) $h(P_j)$ constructs both a path and a cycle with edges of M_i.

3. If $h(P_j)$ contains edges of both $\{M_i \cap M_j\}$ and $\{M_j \setminus \{M_i \cap M_j\}\}$ and

   a) $h(P_j)$ constructs a path with edges of M_i; or

   b) $h(P_j)$ constructs a cycle with edges of M_i; or

   c) $h(P_j)$ constructs a path and a cycle with edges of M_i.

It is obvious that those nine cases are covered by previous five ones, this is because case one and five that are previously verified are a condition for the case $(1.a), (1.b), (1.c)$ and $(3.a), (3.b), (3.c)$, respectively. Simultaneously, case two, three and four that are verified previously can be used to prove the impossibility of $(2.a), (2.b), (2.c)$. And they have been proved to be invalid based on how edges of $h(P_j)$ exist for M_i and M_j. Thus, those nine cases can not be valid as a result of those five ones.

In summary, after focusing on how edges of $h(P_j)$ exist with M_i, because it is concluded that $h(P_j)$ can not exist, P_j can not exist, accordingly. And v_i^- is still not an extra-unmatched node of V_B^- via M_j. Eventually, this theorem is correct. ■

Above all, corollary 8.4 below is concluded to identify each single vertex of V_B^- that could be an unmatched node related to a maximum matching of $B = (V_B, E_B)$ of definition 8.2:

Corollary 8.4
By theorem 8.3 and lemma 8.2, given a fixed and arbitrary maximum matching of $B = (V_B, E_B)$, each unmatched node of V_B^- related to a different maximum matching can be always identified through extra-unmatched node of V_B^- via this given maximum matching.

PROOF According to lemma 8.2, given a fixed and arbitrary maximum matching of B, unmatched nodes with respect to other different maximum matchings can be identified by extra-unmatched nodes via a given maximum matching. Then, by theorem 8.3, all unmatched nodes with respect to all different maximum matchings from this given one can be identified. Therefore, each node of V_B^- that is an unmatched node related to a maximum matching of B can be identified. ∎

Based on the corollary 8.4, algorithm 8.1 thus identifies all single nodes that are unmatched nodes related to maximum matchings of B. In this algorithm, let M_B be a maximum matching of B, and those identified vertices are unmatched nodes with respect to M_B, and extra-unmatched nodes of V_B^- via M_B, respectively. Also, let S_{n_1} be an initially empty node set.

Algorithm 8.1: Find all existing unmatched nodes of V_B^-

Input: $B = (V_B, E_B)$ of definition 8.2, S_{n_1}
Output: Single vertices of V_B^-

```
Identify M_B of B by Hopctoft-Karp algorithm [60];
Set the direction of each edge of M_B from V_B^+ to V_B^-; Set the direction of
  each edge of E_B \ M_B from V_B^- to V_B^+;
Identify unmatched nodes of V_B^- related to M_B;
Add identified nodes into S_{n_1};
while each unmatched node of V_B^- related to M_B do
    Run Breath-First Search (BFS) algorithm [42] from it once to visit
      matched nodes of V_B^- related to M_B ;
    Add visited matched nodes of V_B^- into S_{n_1};
return S_{n_1};
```

PROOF Initially, identifying M_B of B costs $O(\sqrt{|V_B|} \cdot |E_B|)$ at most by running Hopctoft-Karp algorithm [60]. Then, setting directions of edges of E_B in $O(|E_B|)$ time ensures that matched nodes related to M_B can be visited by unmatched nodes via paths alternatively involving edges of M_B and $E_B \setminus M_B$. Obviously, in the **while** loop, BFS algorithm is applied to identify connected nodes of V_B^- and unmatched nodes related to M_B. Also, each visited node of V_B^-, which is matched by M_B, is an *extra-unmatched node of* V_B^- *via* M_B by lemma 8.2 and definition 8.3. By running breath-first search algorithm, each extra-unmatched nodes of V_B^- via M_B can be visited in $O(|E_B| + |V_B|)$ time steps. Finally, time complexity of finding each unmatched node of V_B^- related to maximum matchings of B is $O(\sqrt{|V_B|} \cdot |E_B|)$ in the worst case. ∎

Compared with the method of [62] in the worst-case execution time, which is represented by $O(|V_B| \cdot |E_B|)$, with a bipartite graph mapped by the input network, procedure illustrated by algorithm 8.1 is more efficient. Additionally, according to lemma 8.1 and algorithm 8.1, corollary 8.5 below is concluded to deduce worst-case execution time of identifying nodes of all minimum sets of driver nodes of $D = (V, E)$ is concluded:

Corollary 8.5 (Time complexity of identifying all single driver nodes)
Given $D = (V, E)$ of definition 8.1, the worst-case execution time of identifying nodes of all minimum sets of driver nodes of $D = (V, E)$ is $O(\sqrt{|V|} \cdot |E|)$.

PROOF By definition 8.2, because mapping $D = (V, E)$ of definition 8.1 into $B = (V_B, E_B)$ costs $O(|V| + |E|)$ time, and $|E_B| = |E|$, $2|V| = |V_B|$. According to lemma 8.1 and algorithm 8.1, combining with the worst-case execution time of running algorithm 8.1, identifying nodes of all minimum sets of driver nodes of $D = (V, E)$ is thus $O(\sqrt{|V|} \cdot |E|)$. ∎

8.3.2 Unmatched-node Classification

Given $B = (V_B, E_B)$ of definition 8.2, and $\{v_i^-, v_i^+\} \subseteq V_B$, let v_i^- be an unmatched node with respect to a maximum matching of B. Then, all impacts of removing $\{v_i^-, v_i^+\}$ on unmatched nodes of $V_B^- \setminus v_i^-$ with respect to a maximum matching of $B \setminus \{v_i^-, v_i^+\}$, are concluded by following theorem 8.6:

Theorem 8.6
Given $B = (V_B, E_B)$, let M_i be a maximum matching of B, and S be a set of unmatched nodes of V_B^- related to M_i, where $v_i^- \in S$. Also, when v_i^+ is matched by M_i, let $e = (v_i^+, v_j^-) \in M_i$ and let $P_{v_i^+}$ be a path that leads v_i^+ to be an extra unmatched node of V_B^+ via M_i. Then, in $B \setminus \{v_i^-, v_i^+\}$, one of following cases occurs:

1. *if v_i^+ is a matched node related to M_i and $P_{v_i^+} = \emptyset$. Then, $\{S \setminus v_i^-\} \cup v_j^-$ is a set of all unmatched nodes with respect to the maximum matching $M_i \setminus e$;*
2. *if v_i^+ is either an unmatched node related to M_i, or v_i^+ is matched and $P_{v_i^+} \neq \emptyset$. Then, accordingly, $S \setminus v_i^-$ is a set of unmatched nodes with respect to either M_i or $M_i \oplus P_{v_i^+}$.*

PROOF Since v_i^- is unmatched related to M_i, its removal does not influence both M_i and $S \setminus v_i^-$, maximum matching of $B \setminus \{v_i^-, v_i^+\}$ then only depends on v_i^+.

For the first case, if v_i^+ is a matched node related to M_i, $(v_i^+, v_j^-) \in M_i$ and $P_{v_i^+} = \emptyset$. Then, by lemma 8.2 and theorem 8.3, v_i^+ is also not an extra-unmatched node of V_B^+ related to other different maximum matchings of B. Thus, v_i^+ is contained by all maximum matchings of B, and $M_i \setminus e$ is a maximum matching of $B \setminus \{v_i^-, v_i^+\}$. Further, $\{S \setminus v_i^-\} \cup v_j^-$ is a set of unmatched nodes of $V_B^- \setminus v_i^-$ related to $M_i \setminus e$.

For the second case, when v_i^+ is an unmatched node of V_B^+ with respect to M_i, removal of v_i^+ can therefore not affect existence of M_i. In $B \setminus \{v_i^-, v_i^+\}$, M_i is still a maximum matching and $S \setminus v_i^-$ is a set of unmatched nodes related to it.

On the other hand, if $(v_i^+, v_j^-) \in M_i$ and $P_{v_i^+} \neq \emptyset$, v_i^+ is an extra unmatched node of V_B^+ via M_i by definition 8.3. Then, maximum matching $M_i \oplus P_{v_i^+}$ of $B \setminus v_i^-$ can not be effected by removing v_i^+, and $S \setminus v_i^-$ is still a set of unmatched nodes of $V_B^- \setminus v_i^-$ related to a maximum matching of $B \setminus \{v_i^-, v_i^+\}$. ∎

Based on theorem 8.6, two categories for single unmatched nodes of V_B^- are defined below:

Definition 8.4 (Ordinary & Spare Node)
Given $B = (V_B, E_B)$ of definition 8.2, and $\{v_i^-, {v_i}^+\} \in V_B$, let $v_i^- \in V_B^-$ be an unmatched node related to a maximum matching of B. Then, v_i^- is an ordinary node if and only if case one of theorem 8.6 occurs. Otherwise, v_i^- is a spare node.

Remark 4 Particularly, this definition requires that the node of V_B^- must be an unmatched node related to a maximum matching of B. According to categories of definition 8.4, in B, once any single v_i^- is known as an unmatched node, or an extra-unmatched node of V_B^- via a maximum matching of B, category of v_i^- only depends on if $v_i^+ \in V_B^+$ is a matched node, extra-unmatched node or an unmatched node. □

Nevertheless, for any v_i^+ and v_i^- of $B = (V_B, E_B)$, when they are two *extra-unmatched nodes* of V_B^- and V_B^+ via a same maximum matching, after v_i^- becoming an unmatched node related to another different maximum matching, it is essential to know if v_i^+ is also an *extra-unmatched node* of V_B^+ via this maximum matching. Otherwise, it requires one more time to confirm if v_i^+ is an extra unmatched node of V_B^+ via this maximum matching or not. For this purpose of avoiding unnecessary computation, lemma 8.7 and corollary 8.8 are deduced:

Lemma 8.7
Given $B = (V_B, E_B)$ of definition 8.2, let M_i be a maximum matching of B, $P_{v_i^-}$, $P_{v_j^+}$ be two existing paths, and let $v_i^- \in V_B^-$, $v_j^+ \in V_B^+$ be two extra-unmatched nodes via M_i due to $P_{v_i^-}$ and $P_{v_j^+}$, respectively. Then, $P_{v_j^+}$ still exists in $M_i \oplus P_{v_i^-}$, or vice versa.

PROOF Because $(P_{v_j^+} \oplus M_i)$ and $(M_i \oplus P_{v_i^-})$ are two different maximum matchings of B, and $(P_{v_j^+} \oplus M_i) \oplus (M_i \oplus P_{v_i^-}) = P_{v_j^+} \oplus P_{v_i^-}$, where $(P_{v_j^+} \oplus M_i) \oplus (M_i \oplus P_{v_i^-})$ only results in vertex-disjoint paths or cycles with even length. It is thus that $P_{v_j^+} \oplus P_{v_i^-}$ can only result in an edge set of disjoint paths with even cardinality, which also alternatively involves edges of $P_{v_j^+}$ and $P_{v_i^-}$. Obviously, when $P_{v_j^+}$ and $P_{v_i^-}$ have no common edges, such edge set can be obtained. Otherwise, when $P_{v_j^+}$ and $P_{v_i^-}$ share common edges, $P_{v_j^+} \oplus P_{v_i^-}$ results in disjoint paths with odd cardinality. This is because shared edges by $P_{v_j^+}$ and $P_{v_i^-}$ can be only from M_i, and the number of them is equal to the number of edges out M_i in both $P_{v_i^-}$ and $P_{v_j^+}$. Therefore, there can not be any common edges of both $P_{v_j^+}$ and $P_{v_i^-}$, and $P_{v_j^+}$ therefore still exists in $M_i \oplus P_{v_i^-}$. Similarly, $P_{v_i^-}$ therefore still exists in $M_i \oplus P_{v_j^+}$. ■

Corollary 8.8
According to lemma 8.7, given any pair $\{v_i^-, v_i^+\}$ of V_B, which are two extra-unmatched nodes *of V_B^- and V_B^+ via a same maximum matching. When v_i^- becomes an unmatched node related to another different maximum matching. Then, v_i^+ is an extra unmatched node of V_B^+ via this maximum matching.*

PROOF Based on the proof of lemma 8.7, correctness of this corollary can be proved when it is set that $v_i^+ = v_j^-$ of lemma 8.7. Therefore, after v_i^- becoming an unmatched node related to another different maximum matching, it is valid that v_i^+ is an extra unmatched node of V_B^+ via this maximum matching. ■

Above all, we can further know that if v_i^- is an unmatched node related to a maximum matching of B is independent with if v_i^+ is an unmatched node related to a maximum matching of B, or vice-versa. Therefore, to classify nodes of S_{n_1} returned by algorithm 8.1 into categories of definition 8.4, we just need to identify all extra-unmatched nodes of both V_B^- and V_B^+ via a fixed and arbitrary maximum matching of B once only. Also, it is still required to use algorithm 8.1 to return all vertices of V_B^+ that are unmatched nodes related to maximum matchings of B. Particularly, after line 2 of algorithm 8.1, V_B^- should be replaced with V_B^+. Also, S_{n_1} is replaced with S_{n_2} that is an initially empty node set, and S_{n_2} is the return.

Next, algorithm 8.2 below executes nodal classification of S_{n_1}. Here, let v_i^- be a single vertex of S_{n_1}.

Algorithm 8.2: Classify nodes of S_{n_1}

Input: $B = (V_B, E_B)$ of definition 8.2,S_{n_1}, S_{n_2} returned by using of algorithm 8.1.

Output: Category of each Single vertex of S_{n_1}

```
1 In V_B, colour each node of S_{n_2} into red;
2 while S_{n_1} ≠ ∅ and v_i^- ∈ S_{n_1} do
3 |   S'_{n_1} = S_{n_1} \ v_i^-;
4 |   if v_i^+ is red then
5 |   |   return v_i^- is a spare node;
6 |   |   else if then
7 |   |   |   return v_i^- is an ordinary node;
```

PROOF With S_{n_2} returned by using algorithm 8.1, each node of V_B emerging in it is coloured into red, in order to clearly indicate that this node is an extra-unmatched node of V_B^+ via a maximum matching of B. Then, for each node of S_{n_1}, such as v_i^-, colour on v_i^+ is checked to further confirm category of v_i^- according to theorem 8.6 and definition 8.4. If so, both v_i^- and v_i^+ are unmatched nodes related to a same maximum matching, so that v_i^- is a spare node. Otherwise, v_i^+ is never an extra-unmatched node of V_B^+ related to a maximum matching of B. Thus, case one of theorem 8.6 occurs, and v_i^- is an ordinary node. Because each chosen node of S_{n_1} is removed in line 3, $S_{n_1} = \emptyset$ terminates this algorithm at a moment. For the time complexity, except for obtaining inputs, colouring operations cost $O(|V_B|)$ time, and the **while** loop runs in the same steps at most. The worst-case execution time of this algorithm is $O(|V_B|)$. ■

Last but not the least, by theorem 8.6 and lemma 8.1, a corollary is deduced below, to clarify harmfulness of any single driver-node removal on structurally controlling the residual network:

Corollary 8.9
Given $D = (V, E)$ of definition 8.1, let S_D be a minimum set of driver nodes, and v_i be a single node of S_D. Then, to structurally control $D \setminus v_i$, $\{S_D \setminus v_i\}$ could still be a subset of a minimum set of driver nodes the with cardinality of either $|S_D \setminus v_i|$ or $|S_D|$.

8.3.3 Time Complexity Analysis

Corollary 8.10
*Given $D = (V, E)$ of definition 8.1, the worst-case execution time of solving **Problem 2** is represented by $O(\sqrt{|V|} \cdot |E|)$.*

PROOF As clearly illustrated before, **Problem 2** contains two parts, one is to identify each vertex of V_B^- that is an unmatched node related to a maximum matching of B, and the other is to classify those identified nodes. More specifically, because classifying nodes of V_B^- that are unmatched nodes with respect to maximum matchings of $B = (V_B, E_B)$ requires running algorithm 8.1 twice, and running algorithm 8.2 once. Therefore, given $D = (V, E)$ of definition 8.1, solution of **Problem 2** contains:

1. derive $B = (V_B, E_B)$;
2. run algorithm 8.1 twice;
3. run algorithm 8.2 once.

By definition 8.2, time complexity of mapping D into B is $O(|V| + |E|)$. And the worst-case execution time of solving **Problem 2** is $O(|V| + |E| + \sqrt{|V_B|} \cdot |E_B|)$. Also, due to $2|V| = |V_B|$ and $|E| = |E_B|$, the worst-case execution time is also represented by $O(\sqrt{|V|} \cdot |E|)$. ■

8.4 Summary

To protect structural control into networks with CT-LTI dynamics against control hijack and disruption through driver nodes, and further understand the harmfulness of single driver-node removal, this chapter proposes to efficiently identify each vertex involved into all minimum sets of driver nodes, and classify it by impacts of its removal on the minimum set of driver nodes to structurally control the residual network. Based on the minimum input theorem, this problem is modelled into identifying each node that is an unmatched node related to a maximum matching of the given network, and classifying it by impacts of its removal on the unmatched nodes related to the residual digraph's maximum matching. With a bipartite graph mapped by the input network, this graph problem is eventually solved with the same time complexity as finding a maximum matching of a digraph in total.

By contrast, to identify vulnerable single nodes to the removal, next chapter would extend the work of this chapter and classify all network vertices by impacts of any single-node removal on the structural control into the residual network with a minimum set of inputs.

Chapter 9

Identify Vulnerable Nodes for Network Structural Control

9.1 Overview

According to corollary 8.9 of chapter 8, it is already known that removing any single driver node can not increase the minimum number of driver nodes or inputs to structurally control the residual CT-LTI dynamical networks. However, through related works of section 3.3.1 of chapter 3, continuously removing single network vertices can dramatically increase the minimum number of inputs to structurally control the residual network. By contrast, the driver-node based classification of chapter 8 should be further extended to all general network vertices, essentially.

As shown in section 3.4.3 of chapter 3, many nodal indices have been raised to qualitatively reflect importance of each indexed single vertex in terms of "obtaining" the structural controllability with a minimum set of inputs. But they are difficult to compute for entire network vertices, and none of them can explicitly indicates the importance of any single node in"maintaining" the structural control with a minimum set of inputs. Besides, even some indices of [113] show the importance of single nodes in maintaining structural controllability, an efficient classification scenario is still in lack.

Therefore, this chapter solves the research question 6. With assumptions of section 1.3.2 of chapter 1, a minimum-input structurally controllable network with CT-LTI dynamics is assumed as an input network. This chapter efficiently classifies its all single vertices according to the importance of any single vertex in maintaining the current minimum number of inputs. Besides, the minimum set of inputs is assumed to be obtained by the maximum-matching based method of section 2.4.1 of chapter 2. Also, this input network is also assumed to contain a precomputed maximum matching.

According to corollary 2.5 and theorem 2.8 of chapter 2, because each minimum set of inputs to structurally control a CT-LTI dynamical network is directly adjacent to the same number of unmatched nodes with respect to a maximum matching. Research question of this chapter is thus modelled into a graph-theoretical problem: given the input network, classifying its all single vertices according to the importance of any single vertex in maintaining current unmatched nodes related to a maximum matching. For the solution of this problem, the input network is firstly mapped into a directed bipartite graph. Then, we define the so-called pre-augmenting path of this bipartite graph, in order to confirm all numerical impacts of a single node removal on the number of unmatched nodes of the residual input network. Accordingly, this chapter concludes that a node of the input network is critical, redundant or ordinary, if its removal increases, reduces, or does not change

the initial number of unmatched nodes related to a maximum matching of the input network. To confirm the category of a single node, in a straightforward manner, people can recompute a maximum matching of the network after removing it. However, to classify all vertices of a network with n vertices and m edges in number, the worst-case execution time would be $O(n^{1.5} \cdot m)$ by iterative computation. By contrast, based on the pre-augmenting path, some algorithms that label vertices are designed. As a result, the worst-case execution time of implementing entire nodal classification is $O(m + n)$ steps at most. Also, the space-complexity cost by entire classification is also linear.

For the contribution of this chapter, given a minimum-input structurally controllable CT-LTI dynamical network, it quantitatively explains the surge of the minimum number of inputs to structurally control each residual network during continuous single-vertex removals. Besides, this chapter also efficiently classifies all single network vertices accordingly in linear time and space, so that each vulnerable single vertex to the removal can be explicitly identified.

Remaining chapter is structured as follows: section 9.2 specifies the research question; section 9.3 defines all nodal categories; 9.4 confirms the categories of a single vertex; 9.5 shows algorithms to execute the entire nodal classification; the last section 9.6 summarizes this chapter.

9.2 Problem Formulation

In the beginning, the input network of this chapter is defined:

Definition 9.1 (Input Network of Chapter 9)
Let $D = (V, E)$ be a large and finite digraph, excluding selfloops, parallel arcs and isolated nodes. Also, let M_D be a fixed and arbitrarily precomputed maximum matching by the algorithm of [60], where $V \neq \emptyset$, $V = \{v_i | 1 \leq i \leq n\} (n \geq 3)$, and $E \neq \emptyset$, $|E| \geq 2$, $E = \{\overrightarrow{\langle v_i, v_j \rangle} | i \neq j, v_i, v_j \in V\}$.

Then, assume that this input network and a minimum set of inputs together construct a structurally controllable system, which is represented by following state equation:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{9.1}$$

where, $\mathbf{A} \in \mathbb{R}^{n \times n}$, and $D = (V, E)$ of definition 9.1 only maps into the matrix $\mathbf{A}$. For each arc $\overrightarrow{\langle v_i, v_j \rangle} \in E$, only one non-zero entry of noted by $a_{ji} \in \mathbf{A}$ exists correspondingly. Also, the number of columns of matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ is minimum. With $D = (V, E)$ of definition 9.1, research question of this chapter is formally defined:

Research Question: Given $D = (V, E)$ and M_D. Then, efficiently classify all single nodes of D according to the importance of any single node in maintaining the current minimum set of inputs.

By the maximum-matching based method of section 2.4.1 of chapter 2, or the minimum input theorem 2.8 of section 2.6, because each minimum set of inputs is directly adjacent to a set of unmatched nodes related to a maximum matching of a

given digraph with CT-LTI dynamics. Therefore, research question of this chapter is modelled into a graph-theoretical problem:

Problem 1: Given $D = (V, E)$ and M_D of definition 9.1, classify all single nodes of D according to the importance of any single node in maintaining the current unmatched nodes with respect to a maximum matching of D.

Problem 9.2 is eventually solved in following sections as a way to solve the research question of this chapter. In detail, following chapter mainly defines nodal categories in section 9.3 and then classifies all nodes accordingly in section 9.4 and 9.5.

9.3 Nodal Categories

In this section, three kinds of nodal categories are defined, which can be further used to explicitly distinguish the importance of each involved node in terms of maintaining structural controllability with a minimum set of inputs. Necessarily, in section 9.3.1, a directed bipartite graph of definition 9.2 is used for the reason implied by lemma 9.1. Then, in section 9.3.2, within this bipartite graph, the so-called pre-augmenting path is defined in definition 9.3, in order to further conclude all numerical impacts that are shown in section 9.3.3.

9.3.1 Preliminaries

Definition 9.2

Given $D = (V, E)$ with M_D of definition 9.1, let $B = (V_B, E_B)$ be a directed bipartite graph, V_B^+ and V_B^- be two independent sets of V_B, where $|E_B| = |E|$, $|V_B| = 2|V|$, $|V_B^-| = |V_B^+|$, and $V_B = \{\{v_i^-, v_i^+\} | v_i^- \in V_B^-, v_i^+ \in V_B^+\}$. Besides, let M_B be a maximum matching of B. Then, let $\beta : V \to V_B$, $\gamma : E \setminus M_D \to E_B \setminus M_B$, and $\eta : M_D \to M_B$ be three different bijections. For any $v \in V$, $\beta : v \to \{v^+, v^-\}$, $v^- \in V_B^-, v^+ \in V_B^+$; for any $\overrightarrow{\langle v_i, v_j \rangle} \in E \setminus M_D$, $\gamma : \overrightarrow{\langle v_i, v_j \rangle} \to \overrightarrow{\langle v_i^+, v_j^- \rangle}$, where $\overrightarrow{\langle v_i^+, v_j^- \rangle} \in E_B$; for any $\overrightarrow{\langle v_p, v_q \rangle} \in M_D$, $\eta : \overrightarrow{\langle v_p, v_q \rangle} \to \overrightarrow{\langle v_q^-, v_p^+ \rangle}$, where $\overrightarrow{\langle v_q^-, v_p^+ \rangle} \in M_B$.

Any edge like $\overrightarrow{\langle v_i^+, v_i^- \rangle}$ can not exist in B, otherwise D includes the self loops, and an example of bijections of definition 9.2 is shown by figure 9.1:

In remaining parts of this chapter, let $v \in V$ be a single vertex of $D = (V, E)$ of definition 9.1, based on definition 9.2, removing $v \in V$ in D corresponds to removing $\{v^+, v^-\} \subseteq V_B$ in B, where $\{v^+, v^-\}$ is mapped by v. Similarly to the lemma 8.1 of chapter 8, the following lemma 9.1 illustrates how to use B with M_B to find unmatched nodes of D with respect to M_D:

Lemma 9.1

Given $D = (V, E)$ and M_D of definition 9.1, $B = (V_B, E_B)$ and M_B of definition 9.2, and $\{v^-, v^+\} \subseteq V_B$ mapped from $v \in V$. Then, $v \in V$ in D is an unmatched node with respect to M_D, if and only if $v^- \in V_B^-$ is unmatched with respect to M_B.

PROOF $\Rightarrow$: If $v \in V$ in D is an unmatched node with respect to M_D, and $\{v^-, v^+\} \subseteq V_B$ is mapped from v. By definition 9.2, $v^- \in V_B^-$ can not be the tail of any arc of M_B, but head of arcs of $E_B \setminus M_B$. Thus, v^- is an unmatched node related to M_B in V_B^-.

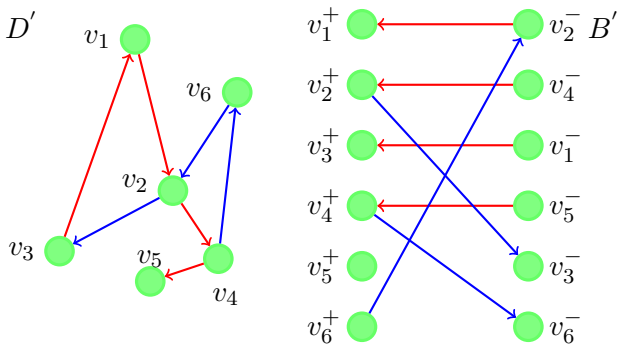

Figure 9.1:

A digraph D' contains a maximum matching that is a red path, which is mapped into a directed bipartite graph B' by definition 9.2 with a maximum matching involving all red arcs.

$\Leftarrow$: If v^- is an unmatched node in V_B^- and related to M_B, in $B = (V_B, E_B)$. v^- is a head of an arc out of M_B. By definition 9.2, let $\overrightarrow{\langle v_i^+, v^- \rangle} \notin M_B$ be such an arc, and there must be $\gamma^{-1} : \overrightarrow{\langle v_i^+, v^- \rangle} \to \overrightarrow{\langle v_i, v \rangle}$, and $\overrightarrow{\langle v_i, v \rangle} \notin M_D$. Thus, v is unmatched related to M_D in D. ■

By lemma 9.1 and definition 9.2, it is true that the number of unmatched nodes of $D \backslash v$ with respect to a maximum matching is the same as that of $V_B^- \backslash v^-$ related to a maximum matching of $B \backslash \{v^-, v^+\}$, where v maps into $\{v^-, v^+\}$ by definition 9.2. Furthermore, numerical impacts of removing v of D on unmatched nodes of $D \setminus v$ are thus equivalent to numerical impacts of removing $\{v^-, v^+\}$ of B on unmatched nodes of $V_B^- \setminus v^-$. Clearly, it needs to know the cardinality of a maximum matching of $B \setminus \{v^-, v^+\}$.

Obviously, if $\{v^-, v^+\} \nsubseteq M_B$, $|M_B|$ must be the cardinality of the maximum matching of $B \backslash \{v^-, v^+\}$. Yet, in another aspect, if there is an arc noted by $\overrightarrow{\langle v^-, v_i^+ \rangle} \in M_B$ or $\overrightarrow{\langle v_j^-, v^+ \rangle} \in M_B$, without recomputation, referring to the augmenting path of definition 2.12 of chapter 2, we would use the path defined in section 9.3.2 below to efficiently confirm the cardinality of a maximum matching of $B \setminus \{v^-, v^+\}$.

9.3.2 Pre-augmenting Path

Definition 9.3 (Pre-augmenting path)
Given $B = (V_B, E_B)$ with M_B of definition 9.2, with respect to M_B, a pre-augmenting path is a path alternatively involving edges of M_B and $E_B \setminus M_B$, and only one of its two terminals is not incident to an edge of M_B(See examples in figure 9.2).

Next, when $\overrightarrow{\langle v^-, v_i^+ \rangle} \in M_B$, the following theorem 9.2 indicates how to use a pre-augmenting path that starts from any node $v^- \in V_B^-$ to confirm the cardinality of the maximum matching of $B \setminus v^-$.

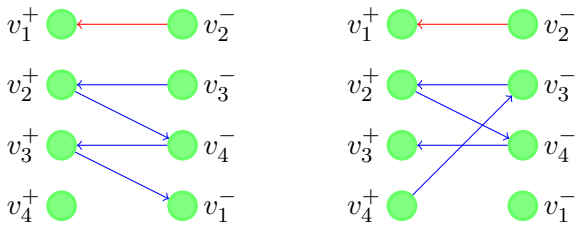

Figure 9.2: Pre-augmenting Paths.

Fig 2. In two bipartite graphs obtained by definition 9.2, with respect to a maximum matching $\{\overrightarrow{\langle v_2^-, v_1^+\rangle}, \overrightarrow{\langle v_3^-, v_2^+\rangle}, \overrightarrow{\langle v_4^-, v_3^+\rangle}\}$, each complete blue path is a *pre-augmenting path*.

Theorem 9.2
In $B = (V_B, E_B)$ of definition 9.2, given any single arc $\overrightarrow{\langle v^-, v_i^+\rangle} \in M_B$. Then, $|M_B|$ is still the cardinality of the maximum matching of $B \setminus v^-$, if and only if v^- is the starting vertex of an existing pre-augmenting path with respect to M_B.

PROOF $\Leftarrow$: Let P be a pre-augmenting path that starts from v^- related to M_B. Then, in $B \setminus v^-$, v_i^+ is not incident to any arc of the matching $M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}$. Thus, by definition 2.12, $P \setminus v^-$ is an augmenting path with respect to $M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}$. By corollary 2.3, we obtain a matching: $\{M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}\} \oplus \{P \setminus v^-\}$ with the cardinality $|M_B|$. Assume that there is at least one augmenting path with respect to $\{M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}\} \oplus \{P \setminus v^-\}$. On the one hand, if this augmenting path excludes arcs of $P \setminus v^-$ and it is related to arcs of M_B, the maximality of M_B is contradicted. On the other hand, if this augmenting path includes arcs of $P \setminus v^-$, an augmenting path would exist in B with respect to M_B, which contradicts the maximality of M_B. Therefore, by corollary 2.4 of chapter 2, $\{M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}\} \oplus \{P \setminus v^-\}$ is a maximum matching in $B \setminus v^-$ with cardinality $|M_B|$.

$\Rightarrow$: If $|M_B|$ is the cardinality of the maximum matching of $B \setminus v^-$, by corollary 2.3 and corollary 2.4 of chapter 2, there should exist an augmenting path with respect to matching $M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}$. Because any two vertices that are not incident to edges of M_B can not be the two terminals of an augmenting path related to $M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}$ at the same time. Otherwise, cardinality of M_B is contradicted. Thus, v_i^+ is the only possible starting vertex of any augmenting path with respect to the matching $M_B \setminus \overrightarrow{\langle v^-, v_i^+\rangle}$, and v^- is therefore the starting vertex of an existing pre-augmenting path with respect to M_B. ■

In a similar way, when $\overrightarrow{\langle v_i^-, v^+\rangle} \in M_B$, we can also conclude that $|M_B|$ is still the cardinality of the maximum matching of $B \setminus v^+$ if and only if v^+ is the ending vertex of an existing pre-augmenting path with respect to M_B. Additionally, the distribution of multiple pre-augmenting paths in $B = (V_B, E_B)$ with respect to M_B is clarified by following theorem:

Theorem 9.3
Given $B = (V_B, E_B)$ and M_B of definition 9.2, and let $v_p^+ \in V_B^+, v_q^- \in V_B^-$ be any two nodes that are not incident to edges of M_B. Also, let $P_{v_p^+}$ be a pre-augmenting path starting

from v_p^+, and let $P_{v_q^-}$ be a pre-augmenting path ending at v_q^-. Then, when $P_{v_p^-}$ and $P_{v_q^+}$ exist simultaneously in B, they are vertex disjoint.

PROOF Assume there is a node shared by $P_{v_p^-}$ and $P_{v_q^+}$ in B. Then, from this shared node, an alternating path involving edges of $E_B \setminus M_B$ and M_B and starting from v_q^+ can be obtained, which is the subpath of $P_{v_q^+}$. Simultaneously, another alternating path involving edges of $E_B \setminus M_B$ and M_B and ending at v_p^- can be also obtained, which is the subpath of $P_{v_p^-}$. As a result, an augmenting path with respect to M_B is constructed by these two alternating paths. By corollary 2.3 of chapter 2, maximality of M_B is thus contradicted. Therefore, when $P_{v_p^-}$ and $P_{v_q^+}$ exist at the same time in B, they must be vertex disjoint. ■

9.3.3 Define Nodal categories

Given $B = (V_B, E_B)$ with M_B of definition 9.2, and any $\{v^+, v^-\} \subseteq V_B$, theorem 9.4 below concludes all numerical impacts of removing $\{v^-, v^+\}$ from B on unmatched nodes of $V_B^- \setminus v^-$ compared with that of V_B^-.

Theorem 9.4
In $B = (V_B, E_B)$, let N_1 be the number of unmatched nodes of V_B^- with respect to a maximum matching of B, and N_2 be the number of unmatched nodes of $V_B^- \setminus v^-$ with respect to a maximum matching of $B \setminus \{v^-, v^+\}$. Also, let d be an integer, and $d = N_2 - N_1$. Then, there must be $d \in \{+1, 0, -1\}$.

PROOF In B, $N_1 = |V_B^-| - |M_B|$. Since there can not be a maximum matching of $B \setminus \{v^-, v^+\}$ with the cardinality bigger than $|M_B|$, the lower limit of N_2 is obtained: $|V_B^- \setminus v^-| - |M_B|$, and the minimum value of d is thus -1. For instance, when $\{v^-, v^+\} \nsubseteq M_B$, M_B is still a maximum matching of $B \setminus \{v^-, v^+\}$. Besides, given $\{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\} \subseteq M_B$, it is possible that $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\}$ is a maximum matching of $B \setminus \{v^-, v^+\}$. For example, if $\overrightarrow{\langle v^-, v_i^+\rangle}$ and $\overrightarrow{\langle v_j^-, v^+\rangle}$ are not adjacent to any other edges. Further, the upper limit of N_2 is thus $|V_B^- \setminus v^-| - |M_B \setminus \{v^-, v^+\}|$, and the maximum value of d is $+1$. Because d is an integer, and it is possible that $d = 0$ in reality. For example, given $\{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\} \subseteq M_B$, by theorem 9.2, v^- is a starting node of a pre-augmenting path, whilst v^+ is not the ending node of any pre-augmenting path. In summary, the value domain of d is $\{+1, 0, -1\}$. ■

Above all, corollary 9.5 is concluded to clarify quantitative impacts of removing any $v \in V$ in $D = (V, E)$ of definition 9.1 on the unmatched nodes of $D \setminus v$.

Corollary 9.5
Given $D = (V, E)$ with M_D of definition 9.1, after removing any $v \in V$ and compared with $|V| - |M_D|$. Then, by definition 9.2, lemma 9.1 and theorem 9.4, the number of unmatched nodes of $D \setminus v$ would increase one, or decrease one, or not change.

According to corollary 9.5, nodal categories are eventually defined to illustrate all quantitative impacts of removing any single node on unmatched nodes of the residual network:

Definition 9.4 (Nodal Categories)
Given $D = (V, E)$ with M_D of definition 9.1, and any single vertex $v \in V$, compared with $|V| - |M_D|$, with respect to a maximum matching of $D \setminus v$, v is classified into one of following categories:

1. *Critical category, if the number of unmatched nodes of $D \setminus v$ is increased by one;*
2. *Redundant category, if the number of unmatched nodes of $D \setminus v$ is reduced by one;*
3. *Ordinary category, if the number of unmatched nodes of $D \setminus v$ does not change.*

According to the minimum input theorem 2.8 of section 2.6, it has been enough to say that critical nodes are the most vulnerable vertices to the single-node removal. In next section, we classify any single vertex of $D = (V, E)$ of definition 9.1 into one of these categories efficiently. And then all nodes of D would be also efficiently classified in section 9.5.

9.4 A Single-Vertex Classification

9.4.1 Solution

Given any single node $v \in V$ of $D = (V, E)$ of definition 9.1, this section classifies it into one of three categories of definition 9.4. Let $\{v^-, v^+\}$ be two vertices of $B = (V_B, E_B)$ of definition 9.2 and they are mapped by $v \in V$ in $D = (V, E)$, based on lemma 9.1, and theorem 9.4, this problem would be solved by confirming the value of $d \in \{+1, 0, -1\}$ after removing $\{v^-, v^+\}$.

Then, the pre-augmenting path of definition 9.3 related to a given maximum matching is used to deduce all necessary and sufficient conditions of $d = +1$, $d = 0$, and $d = -1$ respectively, which are concluded by lemma 9.6, 9.7, and 9.8. With these deduced statements, we can not only avoid recomputing a maximum matching of $B \setminus \{v^+, v^-\}$, but also confirm value of d in linear time. In the following paper, we define $V(M_B) \subseteq V_B$ as a set of vertices incident to edges of M_B.

Lemma 9.6
Given $B = (V_B, E_B)$ with M_B of definition 9.2, and $\{v^-, v^+\} \subseteq V_B$. After removing $\{v^-, v^+\}$, then, $d = -1$ if and only if one of cases holds:

1. $\{v^+, v^-\} \nsubseteq V(M_B)$;
2. *$v^- \in V(M_B)$ or $v^+ \in V(M_B)$, any $v^- \in V(M_B)$ or any $v^+ \in V(M_B)$ must be a terminal of a pre-augmenting path related to M_B.*

PROOF $\Rightarrow$: By theorem 9.4, if $d = -1$, $|M_B|$ is the cardinality of the maximum matching of $B \setminus \{v^-, v^+\}$. On the one hand, M_B could still be a maximum matching of $B \setminus \{v^-, v^+\}$. Thus, in direct, $\{v^-, v^+\} \nsubseteq V(M_B)$. On the other hand, when $|M_B|$ is still the cardinality of the maximum matching of $B \setminus \{v^-, v^+\}$, and M_B can not exist in $B \setminus \{v^-, v^+\}$, by theorem 9.2 and 9.3, with respect to M_B, $v^- \in V(M_B)$ must be the starting vertex of a pre-augmenting path, or $v^+ \in V(M_B)$ must be the ending vertex of a pre-augmenting path.

$\Leftarrow$: If $\{v^-, v^+\} \nsubseteq V(M_B)$, removing $\{v^-, v^+\}$ from B can not influence M_B, M_B is still a maximum matching of $B \setminus \{v^-, v^+\}$, and $d = -1$. Alternatively, if $v^- \in V(M_B)$, or $v^+ \in V(M_B)$. By theorem 9.2, each of them must be a terminal of an existing pre-augmenting path related to M_B. Also, by theorem 9.3, both existing pre-augmenting paths starting from v^- and ending at v^- are vertex disjoint, so that $|M_B|$ as the cardinality of the maximum matching of $B \setminus \{v^-, v^+\}$, and $d = -1$. ∎

Lemma 9.7
Given $B = (V_B, E_B)$ with M_B of definition 9.2, and $\{v^-, v^+\} \subseteq V_B$. After removing $\{v^-, v^+\}$, then $d = +1$ if and only if:

1. *$v^+ \in V(M_B)$ and $v^- \in V(M_B)$ are not connected in B, and the terminal of any pre-augmenting path related to M_B is neither v^+ nor v^-.*

PROOF Assume there is an arc set $\{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\} \subseteq M_B$.

$\Rightarrow$: If $d = +1$, by theorem 9.4, $|M_B| - 2$ is the cardinality of the maximum matching of $B \setminus \{v^-, v^+\}$, and $\{v^-, v^+\} \subseteq V(M_B)$. Clearly, $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\}$ could be such a maximum matching, and there can not be any augmenting paths with respect to it. By theorem 9.2, both v^- and v^+ can not be the terminal of any pre-augmenting path with respect to M_B, in one aspect. Besides, since v_i^- and v_j^+ are both unmatched nodes with respect to matching $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\}$, and they can not construct an augmenting path together related to $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\}$. v^- and v^+ can not be connected through a path starting from v^- in B.

$\Leftarrow$: If both $v^+ \in V(M_B)$ and $v^- \in V(M_B)$ are not terminals of pre-augmenting paths related to M_B, and they are also disconnected in B. Then, a matching $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\}$ is obtained in $B \setminus \{v^-, v^+\}$. Assume that it is not a maximum matching. By corollary 2.3 of chapter 2, at least one augmenting path related to it should exist. For terminals of such augmenting path, both of them can not be out of M_B. Otherwise, maximality of M_B is contradicted. Thus, at least, one terminal should be a vertex incident to an edge of M_B and adjacent to either v^- or v^+. And such vertex can be either v_i^- or v_j^+. Now, reviewing those given conditions above, no augmenting path related to $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+\rangle}, \overrightarrow{\langle v_j^-, v^+\rangle}\}$ exists. By corollary 2.4 of chapter 2, it is a maximum matching of $B \setminus \{v^-, v^+\}$ and $d = +1$. ∎

Lemma 9.8
Given $B = (V_B, E_B)$ with M_B of definition 9.2, and $\{v^-, v^+\} \subseteq V_B$. After removing $\{v^-, v^+\}$, then $d = 0$, if and only if one of following cases holds:

1. *a terminal of any pre-augmenting path related to M_B is never from $\{v^-, v^+\} \subseteq V(M_B)$, and v^- and v^+ are connected;*
2. *a terminal of a pre-augmenting path related to M_B is from $\{v^-, v^+\} \subseteq V(M_B)$, of which, other vertex is not terminal of any pre-augmenting path;*
3. *a pre-augmenting path related to M_B includes $\{v^-, v^+\} \subseteq V(M_B)$;*
4. *either $v^- \in V(M_B)$, $v^+ \notin V(M_B)$ or $v^+ \in V(M_B)$, $v^- \notin V(M_B)$, and no pre-augmenting path related to M_B either starts from $v^- \in V(M_B)$ or ends at $v^+ \in M_B$.*

PROOF Assume there is an arc $\overrightarrow{\langle v^-, v_i^+ \rangle} \in M_B$ if $v^- \in V(M_B)$, and there is an arc $\overrightarrow{\langle v_j^-, v^+ \rangle} \in M_B$ if $v^+ \in V(M_B)$.

$\Rightarrow$: By theorem 9.4, $d = 0$ means that $|M_B| - 1$ is the cardinality of the maximum matching of $B \setminus \{v^-, v^+\}$. Obviously, $M_B \setminus \overrightarrow{\langle v^-, v_i^+ \rangle}$ or $M_B \setminus \overrightarrow{\langle v_j^-, v^+ \rangle}$ can be a valid case, so case four holds. When $\{v^-, v^+\} \subseteq V(M_B)$, matching $M_B \setminus \{\overrightarrow{\langle v^-, v_i^+ \rangle}, \overrightarrow{\langle v_j^-, v^+ \rangle}\}$ is obtained in $B \setminus \{v^+, v^-\}$. By theorem 9.2, one of node of $\{v^-, v^+\} \subseteq V(M_B)$ must be a terminal of a pre-augmenting path related to M_B. Thus case two and three hold. Besides, by lemma 9.7, $v^- \in V(M_B)$ and $v^+ \in V(M_B)$ could be connected through a path, while none of them can be a terminal of a pre-augmenting path related to M_B. Thus, case one holds.

$\Leftarrow$: If one of those four cases is valid, we can obtain the matching with cardinality of $|M_B| - 1$. In particular, we then should prove that such a matching is the maximum in $B \setminus \{v^-, v^+\}$. By lemma 9.7, it can be proved that case one is sufficient to $d = 0$. Also, since other three cases all uses the pre-augmenting path related to M_B, by theorem 9.2, $d = 0$ can be obtained by them in sufficiency. ∎

According to lemma 9.6, 9.7, and 9.8, for any $\{v^-, v^+\} \subseteq V_B$, confirming the value of d needs to investigate if $v^- \in V(M_B)$ and $v^+ \in V(M_B)$ at the beginning. Then, if so, it should be known as well whether or not that $v^- \in V(M_B)$ or $v^+ \in V(M_B)$ is a terminal of a pre-augmenting path with respect to M_B. If not, it should be further confirmed whether $v^- \in V(M_B)$ and $v^+ \in V(M_B)$ are connected through a path starting from v^-, or not. By definition 9.2, M_B is known by mapping M_D of $D = (V, E)$ of definition 9.1, in $O(|E_B|)$ time, and finding a path starting from or ending at any given vertex can be done by the breadth first search or the breadth first search [42] in $O(|V_B| + |E_B|)$ steps at most. Finally, given $D = (V, E)$ with M_D, value of d can be eventually confirmed in linear time $O(|V_B| + |E_B|)$ to classify a single vertex of D.

9.4.2 Expanding

For the problem of classifying all nodes of $D = (V, E)$ with M_D of definition 9.1 into categories of definition 9.4, by lemma 9.1 and theorem 9.4, it can be solved by confirming the value of $d \in \{+1, 0, -1\}$ for each vertex pair $\{v^-, v^+\} \subseteq V_B$ in $B \setminus \{v^-, v^+\}$. Nevertheless, iteratively finding pre-augmenting paths for each $\{v^-, v^+\}$ would result in the time complexity $O((|V_B| + |E_B|)^2)$. However, it is not what this chapter proposes, definitely.

Instead, because multiple nodes of either V_B^- or V_B^+ can exist in same pre-augmenting paths related to M_B. Therefore, we just need to traverse vertices once and identify traversed vertices of $V(M_B)$. Based on this idea, by lemma 9.6 and 9.7, we can confirm if $d = -1$ or $d = 0$ by traversing B once in linear time.

On the other hand, given $\{v^-, v^+\} \subseteq V(M_B)$, which are not terminals of any pre-augmenting paths related to M_B, to confirm value of $d \in \{0, +1\}$, by lemma 9.6 and 9.7, it is essential to know if such v^- and v^+ are connected through a path starting from v^-. By running our label operations shown by algorithm 9.5 on vertices of V_B once in linear time, we can confirm such connectivity between v^- and v^+ by checking label on v^+ in $O(1)$ time. Therefore, excluding deriving M_D, classifying all nodes of D is efficiently executed in linear time as a result.

Above all, section 9.5 gives related algorithms to systematically confirm the value of $d \in \{+1, 0, -1\}$ after removing each $\{v^-, v^+\}$ from B in the worst case.

9.5 Entire Nodal Classification

This section confirms the value of d after removing each $\{v^-, v^+\}$ from $B = (V_B, E_B)$ of definition 9.2, so that category of every node of $D = (V, E)$ of definition 9.1 can be known eventually. The worst-case execution time and space complexity consumed to implement following algorithms are concerned. Moreover, the time complexity of running each algorithm excludes the time cost by deriving inputs. Also, the space arranged for the commands of each algorithm is also not considered. The overview of entire process of solving problem 9.2 is shown by following procedure 9.1. In following all algorithms, let v be a single vertex of D and $\{v^-, v^+\}$ be a pair of nodes of B, which is mapped by v according to the bijection of definition 9.2.

Algorithm 9.1: Overview of confirming category of each node of D

Input: $D = (V, E)$ with M_D of definition 9.1.
Output: Category of each node of D
1 Derive bipartite graph $B = (V_B, E_B)$ with M_B of definition 9.2;
2 Find vertices of pre-augmenting paths with respect to M_B by algorithm 9.2 and 9.3;
3 Confirm value of $d \in \{0, -1\}$ in $B \setminus \{v^-, v^+\}$ by algorithm 9.4;
4 **if** $\exists\{v^-, v^+\}$ *unable to confirm value of* $d \in \{0, -1\}$ **then**
5 Derive directed acyclic graph $G = (V_G, E_G)$ of definition 9.5;
6 Set labels on nodes of V_G by algorithm 9.5;
7 Confirm value of $d \in \{0, +1\}$ in $B \setminus \{v^-, v^+\}$ by algorithm 9.6;

9.5.1 Find vertices of pre-augmenting paths

By definition 9.3, one of terminals of any pre-augmenting path with respect to M_B in $B = (V_B, E_B)$ of definition 9.2 is the vertex not incident to edges of M_B. We call such a vertex the *leading vertex*, and they can be used to visit other nodes of pre-augmenting paths either starting from or ending at it.

Our first algorithm below finds all leading vertices of V_B. Here, let $e = \overrightarrow{\langle v_x^+, v_y^- \rangle} \in E_B \setminus M_B$. And let S_1, S_2 be two initially empty vertex sets.

PROOF Firstly, any $e \in E_B \setminus M_B$ is chosen to obtain a leading vertex incident to it. For e, by definition 9.3, procedure of line 4-7 tests if e is incident to a leading vertex of a pre-augmenting path. If so, one of nodes incident to e is not coloured into red, which are then added into either S_1 or S_2. Thus, this algorithm is correct before the first iteration. Then, another single edge of left $E_B \setminus M_B$ is chosen to find another leading vertex as before. Due to the edge removal in line 3, each chosen edge can be tested once, so that this procedure is also correct during each loop. For the same reason, this algorithm terminates when $E_B \setminus M_B = \emptyset$, and we obtain S_1 and S_2, which store all leading vertices related to M_B. Hence, this algorithm is able to identify all leading vertices with respect to M_B. ∎

Algorithm 9.2: Find all leading vertices of V_B

Input: $B = (V_B, E_B)$ with M_B of definition 9.2, S_1, S_2.
Output: All leading nodes of V_B

```
1 Colour nodes of M_B into red;
2 while E_B \ M_B ≠ ∅ and e ∈ E_B \ M_B do
3 |   E'_B = E_B \ e;
4 |   if v_x^+ is red and v_y^- is not red then
5 |   |   S'_1 = S_1 ∪ v_y^-;
6 |   |   else if v_x^+ is not red and v_y^- is red then
7 |   |   |   S'_2 = S_2 ∪ v_x^+;
8 return S_1; S_2;
```

Corollary 9.9 (Complexity of algorithm 9.2)
Given $B = (V_B, E_B)$ with M_B of definition 9.2, the worst-case execution time and space cost by running algorithm 9.2 is $O(|V_B| + |E_B|)$, both.

PROOF For the running time, colouring nodes of M_B costs $O(|V_B|)$ time, by which, testing if each chosen edge incident to a leading vertex thus costs $O(1)$ time. Also, each chosen edge is tested once because it is removed in line 3, total time to test all edges of $E_B \setminus M_B$ is $O(|E_B|)$. Time complexity is $O(|V_B| + |E_B|)$.

Also, $O(|V_B|)$ space is cost by coloring nodes of M_B. Another space is cost by inputs, S_1 and S_2, which is also $O(|V_B|)$ totally. Space complexity is $O(|V_B|+|E_B|)$.■

Next, given S_1 returned by algorithm 9.2, algorithm 9.3 identifies starting nodes of all pre-augmenting paths with respect to M_B, which end at nodes of S_1. Here, let l_p be a label to indicate that the labelled vertex is a starting node of a pre-augmenting path that ends at a node of S_1. We also let v_y^- be a single vertex of S_1, and P_0 be an initially empty nodal set, while $P(P_0)$ represents a set of nodes of $V_B \setminus P_0$, which are all tails of arcs whose heads are nodes of P_0. Besides, let $\overrightarrow{\langle v_j^*, v_i^* \rangle}$ be an arc of E_B, where $v_i^* \in P_0$, $v_j^* \in P(P_0)$ and $*$ is either $+$ or $-$.

PROOF Firstly, a vertex $v_y^- \in S_1$ is chosen and added into P_0, leading $P_0 = \{v_y^-\}$ and $P(P_0) \neq \emptyset$. Then, the second **while** loop of line 5-14 finds and label all nodes connected with v_y^- via pre-augmenting paths that end at v_y^-. And those visited nodes of V_B^- are finally found and labelled with l_p. In the second **while** loop, each newly visited node is added into P_0 to keep searching through pre-augmenting paths ending at v_y^-, while each node without incoming arcs is also removed from P_0 in line 8 to avoid repetition. Therefore, any node approaching v_y^- via paths can be thus visited and labelled once only, and the second **while** loop terminates when $P(P_0) = \emptyset$ due to line 6. Further, this algorithm is correct prior to the first iteration.

Then, another vertex of left S_1 is chosen and added into P_0, to identify and label all nodes connected with it through pre-augmenting paths ending at it, where this newly-chosen node is the one and the only vertex of P_0 that may be visited by nodes unlabelled after previous iterations via paths. After this, edge traversing starts from it in the remaining B. Besides, although there may be pre-augmenting paths overlapped at vertices of M_B, for any vertex labelled with S_1 previously, the

Algorithm 9.3: Find terminals of pre-augmenting paths

Input: $B = (V_B, E_B)$ of definition 9.2, S_1 returned by algorithm 9.2.
Output: V_B^- after label operations

```
1  Colour nodes of M_B into red;
2  while S_1 ≠ ∅ and v_y^- ∈ S_1 do
3  |  P_0' = P_0 ∪ v_y^-;
4  |  S_1' = S_1 \ v_y^-;
5  |  while P(P_0) ≠ ∅ and v_j^* ∈ P(P_0) do
6  |  |  E_B' = E_B \ ⟨v_j^*, v_i^*⟩;
7  |  |  if v_j^* is the last node adjacent to v_i^* then
8  |  |  |  P_0' = P_0 \ v_i^*;
9  |  |  if v_j^* not labelled with S_1 or l_p then
10 |  |  |  P_0' = P_0 ∪ v_j^*;
11 |  |  |  if v_j^* ∈ V_B^- then
12 |  |  |  |  Label v_j^* with l_p;
13 |  |  |  |  else if v_j^* ∈ V_B^+ then
14 |  |  |  |  |  Label v_j^* with S_1;
15 return V_B^-;
```

second **while** loop has visited all nodes of directed path or pre-augmenting path ending at it, the vertex involved into any pre-augmenting path is labelled with l_p during each iteration. Thus, this algorithm is also correct during each iteration.

Due to line 4, this algorithm terminates when $S_1 = \emptyset$. This algorithm is correct.■

Corollary 9.10 (Complexity of algorithm 9.3)
Given $B = (V_B, E_B)$ of definition 9.2 and S_1 returned by algorithm 9.2, the worst-case execution time and space cost by running algorithm 9.3 is $O(|V_B| + |E_B|)$, both.

PROOF Because of line 6, and line 9,10, any node of V_B can be visited and added into P_0 once at most, traversing all nodes approaching each chosen vertex of S_1 via pre-augmenting paths runs in $O(|E_B|)$ time, and colouring nodes of M_B costs $O(|V_B|)$ time. Time complexity is $O(|V_B| + |E_B|)$ totally.

Besides, space arranged for auxiliary variables is $O(1)$. Since each node of V_B^- is arranged one label at most, label operation costs space $O(|V_B|)$. For P_0, $P(P_0)$ and the return, each of them costs $O(|V_B|)$ space, respectively. In total, space complexity is $O(|V_B| + |E_B|)$, which also contains inputs space. ■

Given $S_2 \neq \emptyset$, algorithm 9.3 can be slightly modified to derive all ending nodes of pre-augmenting paths that start from nodes of S_2. Here, let v_x^+ be any node of S_2, $P(P_0) \subseteq V_B \setminus P_0$ now be a nodal set involving heads of arcs whose tails are nodes of P_0. Then, S_1, v_y^- and V_B^- are replaced with S_2, v_x^+ and V_B^+, where V_B^+ is finally returned and V_B^+ of line 13 is replaced with V_B^-. Also, replace $\overrightarrow{\langle v_j^*, v_i^* \rangle}$ with $\overrightarrow{\langle v_i^*, v_j^* \rangle}$ in line 6 of algorithm 9.3. Because of the correctness of algorithm 9.3, the modified one is correct and run in linear time and space with given inputs.

9.5.2 Confirm value of $d \in \{0, -1\}$

Next, algorithm 9.4 below confirms the value of $d \in \{0, -1\}$ after removing any $\{v^-, v^+\} \in V_B$. Here, let S_3 be an initially empty node set.

Algorithm 9.4: Confirm value of $d \in \{0, -1\}$

Input: V_B^- returned by algorithm 9.3, V_B^+ returned by the modified one
Output: value of d

```
1  while V_B^+ ≠ ∅ and v^+ ∈ V_B^+ do
2      if {v^+, v^-} coloured then
3          if v^+ and v^- labelled with l_p then
4              return d = −1; v of D is Redundant node;
5          else if v^+ or v^- labelled with l_p then
6              return d = 0; v of D is Ordinary node;
7          else if v^+ and v^- not labelled with l_p then
8              S_3' = S_3 ∪ v^-;
9          else if only v^+ coloured or only v^- coloured then
10             if v^- or v^+ labelled with l_p then
11                 return d = −1; v of D is Redundant node ;
12                 else if v^- or v^+ not labelled with l_p then
13                     return d = 0; v of D is Ordinary node;
14         else if {v^+, v^-} not coloured then
15             return d = −1; v of D is Redundant node;
16     V_B^+' = V_B^+ \ v^+;
17 return S_3 ;
```

PROOF Prior to the first iteration, any $v^+ \in V_B^+$ is chosen to confirm the value of $d \in \{0, -1\}$ after removing $\{v^-, v^+\} \in V_B$. Procedure considers two aspects, one is that if $\{v^-, v^+\} \subseteq V(M_B)$ or not, and another is to check if v^-, v^+ are labelled with l_p, respectively. By algorithm 9.3, all nodes of M_B are coloured into red. By theorem 9.3 and algorithm 9.3, any vertex of V_B^- and labelled with l_p is a terminal of a pre-augmenting path related to M_B, this pre-augmenting path must be vertex disjoint with a pre-augmenting path involving a terminal of V_B^+ and labelled with l_p. Once these two aspects are confirmed, value of $d \in \{0, -1\}$ can be finally confirmed and returned according to lemma 9.6, 9.8. Accordingly, by definition 9.4, category of v of D, which maps into $\{v^-, v^+\}$ is also confirmed as well. Nevertheless, by lemma 9.7 and lemma 9.8, in line 7, 8, it is possible that either $d = +1$ or $d = 0$. In this case, v^- is added into S_3, and value of d in this case is confirmed by following algorithms. Hence, this algorithm is correct before the first iteration.

Then, this algorithm is also correct in each iteration, because those two aspects for each vertex pair are concerned by lemma 9.6,9.8 to confirm value of $d \in \{0, -1\}$. Due to vertex removal of V_B^+, this algorithm terminates when $V_B^+ = \emptyset$. Since value of $d \in \{+1, 0, -1\}$ for each $\{v^-, v^+\}$ is returned along with each $v^+ \in V_B^+$, this algorithm is correct. ∎

Corollary 9.11 (Complexity of algorithm 9.4)
Given V_B^- and V_B^+ returned by using algorithm 9.3, the worst-case execution time of algorithm 9.4 is $O(|V_B|)$ and the space complexity is $O(|V_B|)$.

PROOF For the running time, confirming if any $\{v^+, v^-\} \subseteq V(M_B)$ or not can be done in $O(1)$ time, and confirming value of $d \in \{0, -1\}$ for each $\{v^-, v^+\}$ costs $O(1)$ time with existence of l_p. And also each v^+ is concerned only once with v^-. Time complexity is thus $O(|V_B|)$. Also, space complexity is also $O(|V_B|)$, which is arranged for returns, inputs and auxiliary variables in total. ∎

9.5.3 Confirm value of $d \in \{0, +1\}$

Given $B = (V_B, E_B)$ of definition 9.2, by lemma 9.7 and 9.8, for each node $v^- \in S_3$, which is returned by algorithm 9.4, to understand the value of related $d \in \{0, +1\}$, it needs to confirm if v^- and v^+ are connected or not by a directed path starting from v^- in B, where v^+ and v^- are mapped by v of D. For this purpose, we use a digraph defined by following definition 9.5, noted by $G = (V_G, E_G)$:

Definition 9.5 ($G = (V_G, E_G)$)
Given $B = (V_B, E_B)$ of definition 9.2, let $G = (V_G, E_G)$ be a directed acyclic graph and it excludes parallel arcs, and $C = \{c_i | 1 \leq i \leq l\} (l \geq 0)$ be a set of all strongly connected components of B. Also, $V_G = \{v_i^* | v_i^* \in V_B, v_i^* \notin C\} \cup \{v_{c_1} \ldots, v_{c_i}, \ldots v_{c_l}\}$, where any v_{c_i} corresponds to c_i of B. Besides, E_G is a non-empty edge set. E_G excludes edges of E_B that are incident to two vertices of any c_i of B, while any edge of E_G incident to v_{c_i} in G corresponds to an edge of E_B that is incident to a vertex of c_i of B.

By the algorithm of [111], given $B = (V_B, E_B)$ of definition 9.2, C and $G = (V_G, E_G)$ can be obtained in $O(|V_B| + |E_B|)$ time and space in the worst case, both. An example of definition 9.5 is shown in figure 9.3. In the following, algorithm 9.5 sets the label on each vertex of $G = (V_G, E_G)$ of definition 9.5, so that we can then efficiently know if a vertex is connected with any given vertex via directed paths in G by just checking its label. In this algorithm, for convenience of description, we use u_x to represent any vertex of V_G, and let L_{u_x} be the label on u_x. Particularly, L_{u_x} is a single linked list, where $L_{u_x}.head$ is the first element of L_{u_x}, $L_{u_x}.tail$ is a set of last elements, called the tail set of L_{u_x}, while $u_x.next$ is a pointer on a vertex pointed by u_x via an arc of G. Also, let P_0 be an initially empty node set, and u_s be a vertex of P_0 when $P_0 \neq \emptyset$. Besides, let $\delta(u_s)$ be a set of nodes pointed by u_s, and let u_t be a vertex of $\delta(u_s)$, where $\exists \overrightarrow{\langle u_s, u_t \rangle} \in E_G$.

PROOF When the first vertex u_x is chosen in line 1, this algorithm starts to identify and labels all vertices connected with it through paths of G, which is done by the second **while** loop from line 4 to 15. By definition 9.5, such connectivity in G is equivalent with that in B. In detail, after adding u_x into P_0 and setting L_{u_x} in line 3, $P_0 = \{u_x\}$ and $u_x \equiv u_s$. If $\delta(u_x) \neq \emptyset$, the label on a node pointed by u_x through existing paths are set by line 12 and 15. Also, procedure from line 8 to 11 ensures that P_0 only contains vertices of G that are currently pointing other nodes. Particularly, by line 15, for any vertex connected with u_x, its label's tail set is configured to contain $L_{u_x}.tail$, which is then critical to confirm value of d. Thus, given u_x, prior to the first iteration of the second **while** loop, this algorithm

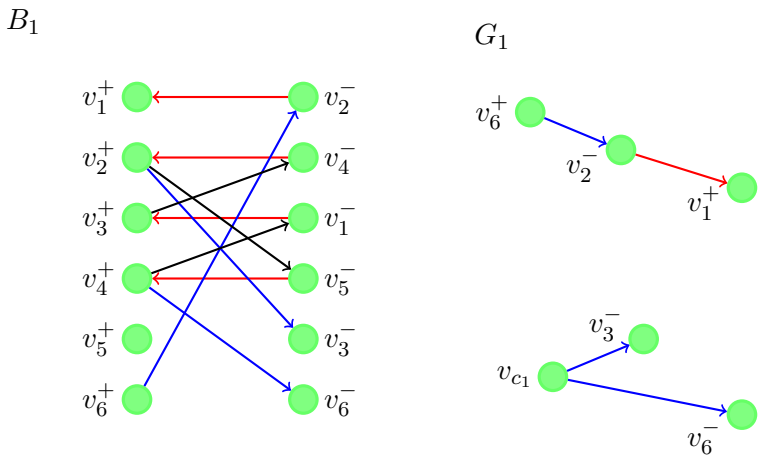

Figure 9.3: An example of definition 9.5.

A bipartite graph B_1 has a strongly connected component: $\{\overrightarrow{\langle v_4^-, v_2^+\rangle}, \overrightarrow{\langle v_2^+, v_5^-\rangle}, \overrightarrow{\langle v_5^-, v_4^+\rangle}, \overrightarrow{\langle v_4^+, v_1^-\rangle}, \overrightarrow{\langle v_1^-, v_3^+\rangle}, \overrightarrow{\langle v_3^+, v_4^-\rangle}\}$. By definition 9.5, a digraph G_1 is obtained.

Algorithm 9.5: Set a label on each node of G

Input: $G = (V_G, E_G)$ of definition 9.5
Output: Labels on vertices of V_G

while $u_x \in V_G$ *not coloured* **do**
  Colour u_x into red **and** $P_0^{'} = P_0 \cup u_x$;
  $L_{u_x}.head = u_x$; $L_{u_x}.tail = \{u_x\}$;
  while $P_0 \neq \emptyset$ ***and*** $\exists u_t \in \delta(u_s)$ **do**
    $E_G^{'} = E_G \setminus \overrightarrow{\langle u_s, u_t\rangle}$;
    if u_t *not coloured* **then**
      Colour u_t into red;
    if u_t *is the last vertex pointed by* u_s **then**
      $P_0^{'} = P_0 \setminus u_s$;
    if $\delta(u_t) \neq \emptyset$ **then**
      $P_0^{'} = P_0 \cup u_t$;
    $L_{u_t}.head = u_t$; $u_t.next = L_{u_s}.head$;
    if $L_{u_t}.tail = \{u_t\}$ **then**
      $L_{u_t}.tail^{'} = L_{u_t}.tail \setminus u_t$;
    $L_{u_t}.tail^{'} = L_{u_t}.tail \cup L_{u_s}.tail$;
return Labels on all nodes of V_G;

is correct. Then, if $P_0 \neq \emptyset$, due to procedure of line 10 and 11, other vertices of P_0 approached by u_x are used to continuously identify and label nodes as before, where the tail set of a label on a node still involves $L_{u_x}.tail$. Hence, this algorithm can identify and label all nodes approached by u_x during iterations of the second **while** loop. Because each vertex of P_0 that has no neighbours pointed by it would be removed in line 9, $P_0 = \emptyset$ terminates the second **while** loop, and all nodes approached by u_x are labelled, whose tail sets must involve $L_{u_x}.tail$.

Next, during each iteration of the first **while** loop, the vertex of G that is not coloured is chosen and added into P_0, which is the only element of P_0 by now. Similarly, the second **while** loop is triggered to identify and label all nodes connected with it through existing paths starting from this newly-chosen node in line 1. In particular, because a node chosen in previous iteration of the first **while** loop might be approached by currently chosen vertex via paths, to guarantee that tail set of this currently chosen node of line 1 is involved by the tail set of each currently traversed single vertex of line 4, procedure of line 13, 14 is thus given. Because each labelled vertex of V_G is coloured in either line 2 or line 7, all vertices of V_G would be coloured into red at a moment, on which this algorithm terminates and all vertices of V_G are labelled. In consequence, this algorithm is correct. ■

Corollary 9.12 (Complexity of algorithm 9.5)
Given $G = (V_G, E_G)$ of definition 9.5, the worst-case execution time and space cost by running algorithm 9.5 is $O(|V_G| + |E_G|)$, both.

PROOF For the running time, it is cost by colouring, visiting and labelling all vertices of V_G. Because each node is coloured once, visited once only, and the label operation on a currently traversed node is triggered by an existing incoming edge of current E_G and line 1. Visiting and colouring vertices of V_G thus costs $O(|V_G|+|E_G|)$ at most. Also, by line 3, 12-15, because each label operation on a single vertex costs $O(1)$ time, and there are $O(|E_B|)$ label operations at most. Therefore, the worst-case execution time of this algorithm is $O(|V_G| + |E_G|)$.

For the space complexity, obviously, the whole linked lists of labels generated by this algorithm contains all vertices of V_G, and the number of labels is $|V_G|$. Also, the number of $next$ pointers is $|E_G|$. Thus, space arranged for labels should be $O(|E_G| + |V_G|)$. Besides, space is still cost by colouring operation, inputs and all auxiliary variables, such as P_0, which cost $O(|V_G|)$ in total. Therefore, space complexity of this algorithm is $O(|E_G| + |V_G|)$. ■

Because $D = (V, E)$ of definition 9.1 is a finite digraph, $B = (V_B, E_B)$ of definition 9.2 is a finite bipartite graph, and so that $G = (V_G, E_G)$ of definition 9.5 is a finite directed acyclic graph. Therefore, after running algorithm 9.5, vertices of V_G that exclude incoming arcs of E_G must be involved into all tail sets of all labels on vertices of V_G. Above all, based on lemma 9.7,9.8, algorithm 9.6 below confirms value $d \in \{0, +1\}$ for all nodes of S_3 returned by algorithm 9.4.

Here, let v^- be a vertex of S_3, c_i, c_j be two strongly connected component of B, v_{c_i}, v_{c_j} be two vertices of G corresponding to c_i and c_j by definition 9.5. Besides, let T be a set of labels on nodes of G that have no incoming edges, and we also define T_0 as a set storing T, and it is possible that $c_i = c_j$. Also, if v^- or v^+ is contained by a strongly component, we assume that this component is c_i or c_j.

Algorithm 9.6: Confirm value of $d \in \{0, +1\}$ for all nodes of S_3

Input: All labels returned by algorithm 9.5, S_3 returned by algorithm 9.4, $B = (V_B, E_B)$ of definition 9.2.

Output: Value of d for each node of S_3

1 Derive T; Find each c_i of B by the algorithm of [111];
2 Label vertices of each c_i with c_i;
3 **while** $S_3 \neq \emptyset$ ***and*** $v^- \in S_3$ **do**
4 $\quad S_3^{'} = S_3 \setminus v^-$;
5 $\quad T_0 = T$; $T = \emptyset$;
6 $\quad$ **if** v^+ ***and*** v^- *not labelled with* c_i **then**
7 $\quad\quad L_{v^-}.tail^{'} = L_{v^-}.tail \cup v^-$;
8 $\quad\quad$ **if** $v^- \in L_{v^+}.tail$ **then**
9 $\quad\quad\quad$ **return** $d = 0$; v of D is Ordinary node;
10 $\quad\quad\quad$ **else if then**
11 $\quad\quad\quad\quad$ **return** $d = +1$; v of D is Critical node;
12 $\quad\quad L_{v^-}.tail^{'} = L_{v^-}.tail \setminus v^-$;
13 $\quad\quad$ **else if** v^- *labelled with* c_i ***and*** v^+ *labelled with* c_j **then**
14 $\quad\quad\quad L_{v_{c_i}}.tail^{'} = L_{v_{c_i}}.tail \cup v_{c_i}$;
15 $\quad\quad\quad$ **if** $v_{c_i} \in L_{v_{c_j}}.tail$ **then**
16 $\quad\quad\quad\quad$ **return** $d = 0$; v of D is Ordinary node;
17 $\quad\quad\quad\quad$ **else if then**
18 $\quad\quad\quad\quad\quad$ **return** $d = +1$; v of D is Critical node;
19 $\quad\quad\quad L_{v_{c_i}}.tail^{'} = L_{v_{c_i}}.tail \setminus v_{c_i}$;
20 $\quad\quad$ **else if then**
21 $\quad\quad\quad L_{v^-}.tail^{'} = L_{v^-}.tail \cup v^-$ **or** $L_{v_{c_i}}.tail^{'} = L_{v_{c_i}}.tail \cup v_{c_i}$;
22 $\quad\quad\quad$ **if** $v^- \in L_{v_{c_j}}.tail$ ***or*** $v_{c_i} \in L_{v^+}.tail$ **then**
23 $\quad\quad\quad\quad$ **return** $d = 0$; v of D is Ordinary node;
24 $\quad\quad\quad\quad$ **else if then**
25 $\quad\quad\quad\quad\quad$ **return** $d = +1$; v of D is Critical node;
26 $\quad\quad\quad L_{v^-}.tail^{'} = L_{v^-}.tail \setminus v^-$ **or**$L_{v_{c_i}}.tail^{'} = L_{v_{c_i}}.tail \setminus v_{c_i}$;
27 $\quad\quad T = T_0$;

PROOF Initially, T can be derived by finding nodes of V_G without incoming edges in $O(|V_G|)$ steps at most. Simultaneously, all strongly connected components of B are identified by the algorithm of [111] with time complexity $O(|V_B| + |E_B|)$, and each vertex of any strongly connected component is labelled with the notation of this component, which costs $O(|V_B|)$ time mostly. After this, any vertex $v^- \in S_3$ is chosen to consider if v^+ is approached by it in B, and related value of $d \in \{0, +1\}$ is confirmed by lemma 9.7, 9.8 eventually. According to line 15 of algorithm 9.5, because the tail set of a label on any given vertex of G involves tail sets of labels on all nodes approaching it via paths, it means that if $L_{v^-}.tail \subseteq L_{v^+}.tail$, v^+ is approached by $v^- \in S_3$. Therefore, with returns of algorithm 9.5, this algorithm considers three cases according to if v^- and v^+ are involved into the strongly connected component, where line 20 is set for the situation when v^- and v^+ are not involved into strongly connected compnent of B at the same time. Further, T is

set as empty and v^- is added into $L_{v^-}.tail$, so that any tail of label on the node approached by v^- only contains v^-. Based on definition 9.5 and algorithm 9.5, because any two vertices of a strongly connected component can visit each other via exisitng paths [42], if v^- is involved into c_i, v_{c_i} is used to identify if it is connected with v^+ via paths of G, where paths also exist in B. Consequently, before the first iteration, this algorithm is correct.

After this, the initially chosen v^- is removed from S_3. During each iteration, a vertex is chosen from the remaining S_3 to confirm value of related d as before. After each iteration, since T and changed tail sets are all restored, this algorithm is correct during each iteration.

Because each element of S_3 is removed from S_3 in line 4, $S_3 = \emptyset$ terminates this algorithm, on which related vale of d for each vertex of S_3 is confirmed respectively. Above all, this procedure is correct. ■

Corollary 9.13 (Complexity of algorithm 9.6)
Given labels returned by algorithm 9.5, S_3 returned by algorithm 9.4, and $B = (V_B, E_B)$ of definition 9.2, the worst-case execution time and space of running algorithm 9.6 is $O(|V_B|+|E_B|)$, respectively.

PROOF For the worst-case execution time, running time is firstly cost by deriving T, identifying all strongly connected components of B and label nodes in line 1, 2. By definition 9.5, $|V_G| \leq |V_B|$, time complexity of such initialization is $O(|V_B|+|E_B|)$. Besides, running time is also cost by executing the **while** loop, reviewing the operation on each vertex of S_3 , which involves checking if v^- and v^+ are involved into strongly connected components, updating tail sets and the final comparison, whose time complexity is thus $O(|V_B|)$. Due to line 15 of algorithm 9.5, we can know that the tail set of each visited node of G only contains entities that are vertices without any incoming arcs, and also, tail sets of different nodes may share common subsets on other vertices. After line 5, those sets exclude any entities until running procedures from line 6, during which, each tail set can contain only one entity at most. As a result, the operation on each vertex of S_3 costs $O(1)$ time steps. In total, time complexity of this algorithm is $O(|V_B| + |E_B|)$.

For the space complexity, space is mainly arranged for T, T_0, all identified strongly connected components, and labels on nodes of strongly connected components, which totally cost space $O(|V_B| + |E_B|)$ at most. For other auxiliary variables and returns, they cost $O(1)$ space. Thus, space complexity of this algorithm is $O(|V_B| + |E_B|)$. ■

9.5.4 Complexity Analysis

Corollary 9.14 (Complexity of solving problem 9.2)
Given $D = (V, E)$ with M_D of definition 9.1, the worst-case execution time cost by solving the problem 9.2 is $O(|V| + |E| + \sqrt{|V||E|})$, where $O(\sqrt{|V||E|})$ is cost by deriving M_D. And the space complexity is $O(|V| + |E|)$.

PROOF Through running those algorithms from 9.2 to 9.6 above, category of each v of $D = (V, E)$ is identified by confirming the value of $d \in \{0, +1, -1\}$ with each $\{v^-, v^+\} \in V_B$ in $B = (V_B, E_B)$ of definition 9.2. In aggregation, the worst-case

execution time of running algorithm 9.1 is the sum of each operations. In detail, deriving B from D, and deriving $G = (V_G, E_G)$ of definition 9.5 from B costs $|V_B| + |E_B|$) steps in total. By algorithms from 9.2 to 9.6, time complexity of comfirming the value of $d \in \{0, +1, -1\}$ with each $\{v^-, v^+\} \in V_B$ in $B = (V_B, E_B)$ is thus $O(|V_B| + |E_B|)$. By definition 9.2, $|V_B| = 2|V|$ and $|E_B| = |E|$, total running time of classifying vertices of $D = (V, E)$ into categories of definition 9.4 is $O(|V| + |E| + \sqrt{|V|}|E|)$, where deriving the maximum matching M_D of $D = (V, E)$ costs $O(\sqrt{|V|}|E|)$ by algorithm of [60].

Similarly, space complexity is also concerned. Total space is arranged for the input network $D = (V, E)$, obtaining $B = (V_B, E_B)$, $G = (V_G, E_G)$, and implementing those five algorithms, which is still the sum of the space arranged for each of them. By definition 9.2, 9.5, space arranged for B and G is $O(|E_B| + |V_B|)$, respectively. Referring the space complexity of previous algorithms, and due to $|V_B| = 2|V|$ and $|E_B| = |E|$, space complexity is also $O(|V| + |E|)$ totally. ■

In summary, problem 9.2 is efficiently solved in linear time and space, except for deriving M_D of D. Furthermore, based on all numerical impacts of removing any single vertex on the minimum set of driver nodes, classifying all nodes of an initially minimum-input structurally controlled CT-LTI network can be done in linear time. As a result, we can thus efficiently identify all vulnerable single vertices to malicious removals or failures.

9.6 Summary

In this chapter, based on all possible numerical impacts of removing a single network vertex on the minimum set of inputs to structurally control a network, three nodal categories are concluded to explicitly identify vulnerable single vertices to malicious removals and failures. According to the minimum input theorem, we firstly investigate all possible impacts by exploring all numerical impacts of a single vertex removal on the minimum number of unmatched nodes of a residual digraph. Then, accordingly, within a bipartite graph mapped by the input network, we classify all vertices of a given system network into them by identifying pre-augmenting paths and confirming connectivity between some pairwise vertices that is completed by some label operations. As a result, excluding pre-computing the minimum set of inputs of the input network, the entire nodal classification can be executed in linear time and space.

Part IV

Epilogue

Chapter 10

Conclusion & Future Work

After systematized narration of previous parts, this chapter summaries them, and then illustrates other different things to guide the future research work.

10.1 Thesis Summary

To effectively maintain controllability of continuous time and linear time-invariant (CT-LTI) dynamic systems, this thesis concentrates on efficiently maintaining structural controllability with a minimum set of inputs. It solves six research questions, which are systematically introduced and defined with various motivations in chapter 1 of part I. In general, given a (minimum-input) structurally controllable system or network with CT-LTI dynamics, as the input. Then, main contributions of this thesis are summarized below:

1. Efficient recovery of structural controllability with a minimum set of inputs after very limited and severe modification, respectively. Particularly, very limited modification means either removing or adding a system component, and inputs are strictly constrained for the recovery against severe modification.

2. Efficient network analysis that accurately identifies vulnerable single vertices and edges to the removal, according to the importance of single node and edge in terms of maintaining structural control with a minimum set of inputs.

Viewing these two contributions, six questions are classified into two aspects correspondingly. In detail, the first aspect contains previous three research questions, which are addressed in part II and focus on efficient recovery of the structural controllability with a minimum set of inputs to against very limited and severe modification on an initially minimum-input structurally controllable system. After this, the second aspect contains the last three research questions, which are addressed in part III and concentrate on the efficient network analysis to explicitly identify vulnerable single vertices and edges to the removal.

More specifically, in chapter 2, some knowledge and technologies are clearly presented and compared, which are indispensable to define, analyse and solve those six research questions and our future works. This chapter mainly describes and explains what the controllability is, and how to acquire controllability through the structural controllability with graph-theoretical methods, by which, controllability of complex networks is also discussed. During the description, those two graph-theoretical methods that can effectively derive structural controllability with a minimum set of inputs are systematically shown. Besides, the relationship among

controllability, structural controllability and strongly structural controllability is compared as well. Additionally, from the perspective of graph theory and algebra, the development of the study on acquiring controllability of CT-LTI systems over the last half century could be also directly appreciated in this chapter.

Based on the known knowledge to acquire structural controllability with a minimum set of inputs, chapter 3 scrutinises works related to all research questions. Those reviewed works are about structural-control recovery, robustness of network structural controllability and network analysis to maintain structural controllability. Besides, related graph-theory problems that can model and solve some research questions are also reviewed as well. In the most case, in this chapter, aiming at each research question, existing issues of related works are precisely indicated to justify why those six research questions are raised and addressed in part II and III.

After that, well solving each research question formally starts from chapter 4 and ends at chapter 9 in order. Particularly, for questions about recovery of structural controllability with a minimum set of inputs, it is always assumed that the given system or network with CT-LTI dynamics is initially structurally controllable by a minimum set of inputs, and such inputs are identified through the maximum-matching based method of section 2.4.1 of chapter 2. Besides, each research question is modelled into a graph-theoretical problem in chapter 4, 5 and 6, respectively. As a result of solving these research questions in part II, compared with existing works shown in section 3.2 of chapter 3, the worst-case execution time of our every recovery scenario is more efficient and our solutions contain less constrains. From those results, it is reflected that structural-controllability recovery should not only consider the execution efficiency, but also concern the amount of modification on the given CT-LTI system or network and constrains on the input to recover.

In the following, for questions solved in part III, according to impacts of a single edge or vertex removal on the minimum set of inputs to structurally control the residual network, they are modelled into problems about classifying either single edges in chapter 7 or vertices in 8 and 9. Specifically, chapter 8 focuses on single vertices that could be contained by a minimum set of driver nodes, while chapter 9 considers all kinds of single nodes of the given network. As a result, the worst-case execution time of each solution is the same as the running time of identifying a maximum matching at most. Additionally, due to more efficient average-case time complexity of identifying a maximum matching of an ER random graph, once the input network of solution of each research question is ER random model and sparse, those network analysis might be further executed with lower average-case time complexity.

Furthermore, with those efficient analysis methods and nodal categories, for the study on robustness of network structural controllability against the nodal removal, the surge of the minimum number of inputs to structurally control the residual network can be better explained than using global network properties, such as vertex average degree, vertex betweenness and so on. In comparison of related works about robustness of network structural controllability shown in section 3.3 and network analysis for structural controllability in section 3.4 of chapter 3, it is to say that study on recognising how each single vertex maintains the structural controllability with a minimum set of inputs can be more practically addressed through related graph-theoretical problems.

10.2 The Future Work

In this section, to address limitations of our work, the research still undertaken in the future is illustrated. Generally speaking, following those illustrated research method in previous chapters, our future work still rely on graph-theoretical problems to model and solve corresponding research questions out of this thesis.

10.2.1 Further Study of Maximum Matching

Throughout this thesis, we can directly feel how the significant role of the maximum matching plays in promoting research on CT-LTI system controllability and solving those six research questions. And the key to those results is the efficiency of existing maximum-matching identification algorithms. Over the last half century, although there have not been a novel scenario to improve the worst execution time of identifying a maximum matching for general networks, it is still meaningful to explore improvements through any possible clues. For example, given a general digraph as an input, reviewing the output of a maximum (cardinality) matching algorithm, it is a set of disjoint cycles or paths. Nevertheless, because the breadth-first and depth-first search on a digraph can also produce disjoint cycles and paths. In order to identify a maximum matching, an interesting problem is raised here: how to use those disjoint paths and cycles returned by breadth-first or depth-first search to construct a maximum matching of a given digraph with lower time complexity.

10.2.2 Study of Strongly Structural Controllability

In section 2.5 of chapter 2, we have known the strongly s-controllability through a graph theoretical manner . For any given CT-LTI system, strongly s-cntrollability is a powerful system property to acquire controllability without to know exact values of non-zero entries of both state and input matrices. Also, according to [33], because it might be possible that attackers maliciously infiltrate interactions among system components or network edge, where the value of some non-zero entries of input or state matrix are changed, so that system might be no longer controllable as a result of that malicious infiltration. For such attacks, compared with our current research, because there is no physical damage but exact parameters, the strongly s-controllability is obviously desirable to acquire as soon as possible.

To do this, we would like to consider how to acquire strongly s-controllability with lower time complexity than the existing result of [32]. By theorem 2.6 of chapter 2, it is already known that acquiring strongly s-controllability should obtain a constrained matching of bipartite graph $B_A = (V_A^+ \cup V_A^-, E_A)$ of definition 2.20 and a constrained self-less matching of bipartite graph $B_{A_X} = (V_{A_X}^+ \cup V_{A_X}^-, E_{A_X})$ of definition 2.24, respectively. More specifically, by definition 2.22, a constrained $\{V_{sub}^-\}$-less *(n - m)-matching* in "B_{A_X}" $\setminus V_{sub}^-$ is actually a constrained self-less *(n - m)-matching* in "B_A" $\setminus V_{sub}^-$. This is because a constrained $\{V_{sub}^-\}$-less *(n - m)-matching* in $\{$"B_{A_X}" $\setminus V_{sub}^-\}$ excludes any edge like (v_i^-, v_i^+), and it is the only one matching whose edges incident to $V_A^- \setminus V_{sub}^-$.

Above all, compared with a general maximum matching and a constrained self-less matching within a same bipartite graph, a constrained self-less matching can be obtained by eliminating vertices contained by cycles that alternatively involves

edges of this general maximum matching and edges out of it. From this aspect, it can be explained why the cardinality of a constrained self-less maximum matching is less than that of a general maximum matching. Therefore, the inputs for a strongly s-controllable system might be more than that of a structurally controllable system.

Also, the alternating-cycle matching of 7.2 of chapter 7 could be used to derive a constrained self-less matching, because the existence of any single alternating-cycle matching would generate two matchings with the same vertex sets, which is a contradiction with the constrained matching. Therefore, it is effective to acquire a constrained self-less matching by removing nodes of identified alternating-cycle matching related to a given maximum matching. And the finally resulting matching is thus a constrained self-less matching. As mentioned previously, since a strongly s-controllable system might require more number of inputs than a structurally controllable system, derive the strongly s-controllability with a minimal number of inputs is desirable. Therefore, we would address the problem about identifying a constrained self-less matching through a given maximum matching of bipartite graph $B_{A_X} = (V^+_{A_X} \cup V^-_{A_X}, E_{A_X})$ of definition 2.24.

Additionally, for the purpose of optimizing strongly srtuctural control, we still purpose to approach a constrained self-less matching with the maximum cardinality in the future. Although identifying a maximum constrained self-less matching is said to a NP-complete problem [31], it is still valuable to explore good results with some reasonable assumptions.

10.2.3 Further Network Analysis

With existing works in part III, we have used several nodal categories and efficient nodal classification to explicitly identify vulnerable single vertices to the removal. Even though, focusing on how a single vertex maintains the current structural controllability should be further developed. When combining related works of section 3.2.2 of chapter 3, which focus on robustness of network structural controllability with a minimum set of inputs, it can be clearly found that different kinds of nodal removals can result various kinds of surges of the minimum set of inputs to structurally control the residual network. It means that the removed nodes can influence the category distribution of remaining nodes. Therefore, it is interesting to explore how each single vertex influences vertices of other categories after removing it, such as the number of other category nodes. And the network analysis is developed from single isolated vertex to single related nodes, by which, the robustness against continuous single node removals can be more precisely estimated. Similarly, we would also do the same analyse on network edges, to investigate how a single edge influences other category edges after removing it. Additionally, to sufficiently develop network analysis for the purpose of maintaining network structural controllability with a minimum set of inputs, those global graph properties, such as vertex degree, betweenness, and so on, would be used to assist corresponding investigation and distinguish for related impacts.

In other words, this future work proposes to explore harmful edge or vertex removal scenarios according to global graph properties. By contrast, in the given time slots, such network analysis can better explain the continuous change of the increase of the minimum number of inputs during edge or nodal removals.

Bibliography

[1] AAZAMI, A., AND STILP, K. Approximation algorithms and hardness for domination with propagation. *SIAM Journal on Discrete Mathematics 23*, 3 (2009), 1382–1399. 33, 40

[2] AGGARWAL, C., HE, G., AND ZHAO, P. Edge classification in networks. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on* (2016), IEEE, pp. 1038–1049. 45

[3] AGRAWAL, P., GARG, V. K., AND NARAYANAM, R. Link label prediction in signed social networks. In *IJCAI* (2013). 45

[4] ALBERT, R., AND BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics 74*, 1 (2002), 47. 8, 37

[5] ALBERT, R., JEONG, H., AND BARABÁSI, A.-L. Error and attack tolerance of complex networks. *nature 406*, 6794 (2000), 378–382. 42

[6] ALCARAZ, C., MICIOLINO, E. E., AND WOLTHUSEN, S. Structural controllability of networks for non-interactive adversarial vertex removal. In *International Workshop on Critical Information Infrastructures Security* (2013), Springer, pp. 120–132. 6, 38, 43

[7] ALCARAZ, C., AND WOLTHUSEN, S. Recovery of structural controllability for control systems. In *International Conference on Critical Infrastructure Protection* (2014), Springer, pp. 47–63. 9, 40

[8] ALT, H., BLUM, N., MEHLHORN, K., AND PAUL, M. Computing a maximum cardinality matching in a bipartite graph in time o (n 1.5 mlog n). *Information Processing Letters 37*, 4 (1991), 237–240. 27

[9] ALWASEL, B., AND WOLTHUSEN, S. D. Reconstruction of structural controllability over erdős-rényi graphs via power dominating sets. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference* (2014), ACM, pp. 57–60. 40, 61

[10] ALWASEL, B., AND WOLTHUSEN, S. D. Recovering structural controllability on erdős-rényi graphs via partial control structure re-use. In *International Conference on Critical Information Infrastructures Security* (2014), Springer, pp. 293–307. 40

[11] ALWASEL, B., AND WOLTHUSEN, S. D. Recovering structural controllability on erdős-rényi graphs in the presence of compromised nodes. In *International Conference on Critical Information Infrastructures Security* (2015), Springer, pp. 105–119. 9, 40

[12] ALWASEL, B., AND WOLTHUSEN, S. D. Structural controllability analysis via embedding power dominating set approximation in erdhos-rènyi graphs. In *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on* (2015), IEEE, pp. 418–423. 6, 40

[13] AZRAN, A. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 49–56. 49

[14] BANG-JENSEN, J., AND GUTIN, G. Z. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008. vii, 26

[15] BARABÁSI, A.-L., AND ALBERT, R. Emergence of scaling in random networks. *science 286*, 5439 (1999), 509–512. 8, 37

[16] BAST, H., MEHLHORN, K., SCHAFER, G., AND TAMAKI, H. Matching algorithms are fast in sparse random graphs. *Theory of Computing Systems 39*, 1 (2006), 3–14. 27

[17] BAST, H., MEHLHORN, K., SCHAFER, G., AND TAMAKI, H. Matching algorithms are fast in sparse random graphs. *Theory of Computing Systems 39*, 1 (2006), 3–14. 61, 75

[18] BASWANA, S., GUPTA, M., AND SEN, S. Fully dynamic maximal matching in o(\logn) update time. *SIAM Journal on Computing 44*, 1 (2015), 88–113. 41

[19] BAY, J. Fundamentals of linear state space systems, ser. *Elect. Eng. New York: McGraw-Hill* (1998). vii, 5, 20

[20] BERGE, C. Two theorems in graph theory. *Proceedings of the National Academy of Sciences 43*, 9 (1957), 842–844. 26, 27

[21] BERNSTEIN, A., AND STEIN, C. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms* (2016), Society for Industrial and Applied Mathematics, pp. 692–711. 41

[22] BHAGAT, S., CORMODE, G., AND MUTHUKRISHNAN, S. Node classification in social networks. In *Social network data analytics*. Springer, 2011, pp. 115–148. 49

[23] BHATTACHARYA, S., HENZINGER, M., AND NANONGKAI, D. Fully dynamic approximate maximum matching and minimum vertex cover in o (log3 n) worst case update time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), SIAM, pp. 470–489. 41

[24] BINKELE-RAIBLE, D., AND FERNAU, H. An exact exponential time algorithm for power dominating set. *Algorithmica 63*, 1-2 (2012), 323–346. 33

[25] BOLLOBÁS, B., BORGS, C., CHAYES, J., AND RIORDAN, O. Directed scale-free graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (2003), Society for Industrial and Applied Mathematics, pp. 132–139. 42

[26] BOYER, S. A. *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009. 6

[27] CÁRDENAS, A. A., AMIN, S., LIN, Z.-S., HUANG, Y.-L., HUANG, C.-Y., AND SASTRY, S. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security* (2011), ACM, pp. 355–366. 6

[28] CÁRDENAS, A. A., AMIN, S., AND SASTRY, S. Research challenges for the security of control systems. In *HotSec* (2008). 6

[29] CARDENAS, A. A., AMIN, S., AND SASTRY, S. Secure control: Towards survivable cyber-physical systems. In *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on* (2008), IEEE, pp. 495–500. 6, 7

[30] CARVALHO, M. H. D., ET AL. An o (ve) algorithm for ear decompositions of matching-covered graphs. *ACM Transactions on Algorithms (TALG) 1*, 2 (2005), 324–337. 46

[31] CHAPMAN, A. Strong structural controllability of networked dynamics. In *Semi-Autonomous Networks*. Springer, 2015, pp. 135–150. 5, 34, 36, 140

[32] CHAPMAN, A., AND MESBAHI, M. On strong structural controllability of networked systems: A constrained matching approach. 34, 36, 139

[33] CHAPMAN, A., AND MESBAHI, M. Security and infiltration of networks: A structural controllability and observability perspective. In *Control of Cyber-Physical Systems*. Springer, 2013, pp. 143–160. 9, 139

[34] CHAPMAN, A., AND MESBAHI, M. Security and infiltration of networks: A structural controllability and observability perspective. In *Control of Cyber-Physical Systems*. Springer, 2013, pp. 143–160. 46, 103

[35] CHARTRAND, G., LESNIAK, L., AND ZHANG, P. *Graphs & digraphs*. CRC Press, 2010. vii, 26

[36] CHEN, T., AND ABU-NIMEH, S. Lessons from stuxnet. *Computer 44*, 4 (2011), 91–93. 7

[37] CHEN, X., PEQUITO, S., PAPPAS, G. J., AND PRECIADO, V. M. Minimal edge addition for network controllability. *IEEE Transactions on Control of Network Systems* (2018). 8, 15, 16, 41, 77, 88

[38] CHERIYAN, J. Randomized o(m(—v—)) algorithms for problems in matching theory. *SIAM Journal on Computing 26*, 6 (1997), 1635–1655. 46

[39] CHIANG, K.-Y., NATARAJAN, N., TEWARI, A., AND DHILLON, I. S. Exploiting longer cycles for link prediction in signed networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), ACM, pp. 1157–1162. 45

[40] CHIN, S. P., COHEN, J., ALBIN, A., HAYVANOVYCH, M., REILLY, E., BROWN, G., AND HARER, J. A mathematical analysis of network controllability through driver nodes. *IEEE Transactions on Computational Social Systems 4*, 2 (2017), 40–51. 9, 47

[41] COMMAULT, C., AND VAN DER WOUDE, J. A classification of nodes for structural controllability. 47, 48

[42] CORMEN, T. H. *Introduction to algorithms*. MIT press, 2009. 41, 46, 83, 110, 123, 132

[43] COSTA, M.-C. Persistency in maximum cardinality bipartite matchings. *Operations Research Letters 15*, 3 (1994), 143–149. 46

[44] CSERMELY, P., KORCSMÁROS, T., KISS, H. J., LONDON, G., AND NUSSINOV, R. Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review. *Pharmacology & therapeutics 138*, 3 (2013), 333–408. 38

[45] DAVISON, E. J. Connectability and structural controllability of composite systems. *Automatica 13*, 2 (1977), 109–123. 22, 29

[46] DELPINI, D., BATTISTON, S., RICCABONI, M., GABBI, G., PAMMOLLI, F., AND CALDARELLI, G. Evolution of controllability in interbank networks. *Scientific reports 3* (2013), 1626. 8, 38

[47] DING, J., AND LU, Y.-Z. Control backbone: An index for quantifying a node s importance for the network controllability. *Neurocomputing*, 153 (2015), 309–318. 48

[48] DING, J., LU, Y.-Z., AND CHU, J. Recovering the controllability of complex networks. *IFAC Proceedings Volumes 47*, 3 (2014), 10894–10901. 9, 39

[49] DION, J.-M., COMMAULT, C., AND VAN DER WOUDE, J. Generic properties and control of linear structured systems: a survey. *Automatica 39*, 7 (2003), 1125–1144. 21

[50] DOOSTMOHAMMADIAN, M., AND KHAN, U. A. On the genericity properties in distributed estimation: Topology design and sensor placement. *IEEE Journal of Selected Topics in Signal Processing 7*, 2 (2013), 195–204. 37

[51] ERDDS, P., AND R&WI, A. On random graphs i. *Publ. Math. Debrecen 6* (1959), 290–297. 27, 42

[52] FEIGE, U. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM) 45*, 4 (1998), 634–652. 29

[53] GOH, K.-I., KAHNG, B., AND KIM, D. Universal behavior of load distribution in scale-free networks. *Physical Review Letters 87*, 27 (2001), 278701. 43

[54] GOLUMBIC, M. C., HIRST, T., AND LEWENSTEIN, M. Uniquely restricted matchings. *Algorithmica 31*, 2 (2001), 139–154. vii, 34, 35, 36

[55] GROSS, J. L., YELLEN, J., AND ZHANG, P. *Handbook of graph theory*. Chapman and Hall/CRC, 2013. vii, 29

[56] GUO, J., NIEDERMEIER, R., AND RAIBLE, D. Improved algorithms and complexity results for power domination in graphs. *Algorithmica 52*, 2 (2008), 177–202. 40

[57] GUPTA, M., AND PENG, R. Fully dynamic (1+ e)-approximate matchings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on* (2013), IEEE, pp. 548–557. 27

[58] HADDAD, W. M., AND CHELLABOINA, V. *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton University Press, 2011. 4

[59] HAYNES, T. W., HEDETNIEMI, S. M., HEDETNIEMI, S. T., AND HENNING, M. A. Domination in graphs applied to electric power networks. *SIAM Journal on Discrete Mathematics 15*, 4 (2002), 519–529. vii, 6, 29, 30, 33, 39, 43

[60] HOPCROFT, J. E., AND KARP, R. M. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing 2*, 4 (1973), 225–231. vii, 6, 26, 27, 28, 54, 64, 75, 81, 86, 87, 92, 110, 116, 133

[61] JIA, T., AND BARABÁSI, A.-L. Control capacity and a random sampling method in exploring controllability of complex networks. *Scientific reports 3* (2013). 48

[62] JIA, T., LIU, Y.-Y., CSÓKA, E., PÓSFAI, M., SLOTINE, J.-J., AND BARABÁSI, A.-L. Emergence of bimodality in controlling complex networks. *Nature communications 4* (2013). 8, 9, 12, 17, 46, 47, 48, 103, 104, 110

[63] KALMAN, R. On the general theory of control systems. *Automatic Control, IRE Transactions on 4*, 3 (1959), 110–110. i, 4

[64] KALMAN, R. E. Canonical structure of linear dynamical systems. *Proceedings of the National Academy of Sciences 48*, 4 (1962), 596–600. 19

[65] KALMAN, R. E. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control 1*, 2 (1963), 152–192. 3, 4, 19

[66] KEPHART, J. O., AND CHESS, D. M. The vision of autonomic computing. *Computer*, 1 (2003), 41–50. 4

[67] KNEIS, J., MÖLLE, D., RICHTER, S., AND ROSSMANITH, P. Parameterized power domination complexity. *Information Processing Letters 98*, 4 (2006), 145–149. vii, 30

[68] KUHN, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97. 42

[69] LEE, E. A. Cyber physical systems: Design challenges. In *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on* (2008), IEEE, pp. 363–369. 6

[70] LESKOVEC, J., HUTTENLOCHER, D., AND KLEINBERG, J. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 641–650. 45

[71] LIBEN-NOWELL, D., AND KLEINBERG, J. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology 58*, 7 (2007), 1019–1031. 45

[72] LIN, C. T. Structural controllability. *Automatic Control, IEEE Transactions on 19*, 3 (1974), 201–208. i, vii, xi, 5, 15, 19, 21, 22, 23

[73] LIU, Y.-Y., SLOTINE, J.-J., AND BARABÁSI, A.-L. Controllability of complex networks. *Nature 473*, 7346 (2011), 167–173. xi, 4, 6, 8, 11, 21, 37, 38, 45, 91, 103

[74] LIU, Y.-Y., SLOTINE, J.-J., AND BARABÁSI, A.-L. Control centrality and hierarchical structure in complex networks. *Plos one 7*, 9 (2012), e44459. 47, 48

[75] LOMBARDI, A., AND HÖRNQUIST, M. Controllability analysis of networks. *Physical Review E 75*, 5 (2007), 056110. 4, 8

[76] LOPEZ, J., SETOLA, R., AND WOLTHUSEN, S. *Critical Infrastructure Protection: Advances in Critical Infrastructure Protection: Information Infrastructure Models, Analysis, and Defense*, vol. 7130. Springer Science & Business Media, 2012. 7

[77] LU, Z.-M., AND LI, X.-F. Attack vulnerability of network controllability. *PloS one 11*, 9 (2016), e0162289. 38, 42, 43, 44

[78] LVLIN, H., SONGYANG, L., JIANG, B., AND LIANG, B. Enhancing complex network controllability by rewiring links. In *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on* (2013), IEEE, pp. 709–711. 9

[79] MAYEDA, H. On structural controllability theorem. *IEEE Transactions on Automatic Control 26*, 3 (1981), 795–798. xi, 23

[80] MAYEDA, H., AND YAMADA, T. Strong structural controllability. *SIAM Journal on Control and Optimization 17*, 1 (1979), 123–138. 5, 34

[81] MICALI, S., AND VAZIRANI, V. V. An o (v— v— c— e—) algoithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on* (1980), IEEE, pp. 17–27. 27, 28

[82] MO, Y., AND SINOPOLI, B. Secure control against replay attacks. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on* (2009), IEEE, pp. 911–918. 6

[83] MOHOLKAR, A. V. Security for cyber-physical systems. *International Journal of Computing and Technology 1*, 6 (2014). 7

[84] MOTTER, A. E., AND LAI, Y.-C. Cascade-based attacks on complex networks. *Physical Review E 66*, 6 (2002), 065102. 43

[85] NEVILLE, J., AND JENSEN, D. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data* (2000), pp. 13–20. 49

[86] NEWMAN, M. *Networks: an introduction*. OUP Oxford, 2010. 27

[87] NEWMAN, M. E. The structure and function of complex networks. *SIAM review 45*, 2 (2003), 167–256. 8, 37

[88] NIE, S., WANG, X., ZHANG, H., LI, Q., AND WANG, B. Robustness of controllability for networks based on edge-attack. *PloS one 9*, 2 (2014), e89066. 9, 44, 53

[89] NORMAN, R. Z., AND RABIN, M. O. An algorithm for a minimum cover of a graph. *Proceedings of the American Mathematical Society 10*, 2 (1959), 315–319. 26, 27

[90] ONAK, K., AND RUBINFELD, R. Maintaining a large matching and a small vertex cover. In *Proceedings of the forty-second ACM symposium on Theory of computing* (2010), ACM, pp. 457–464. 41

[91] PEQUITO, S., KAR, S., AND AGUIAR, A. P. A structured systems approach for optimal actuator-sensor placement in linear time-invariant systems. In *American Control Conference (ACC), 2013* (2013), IEEE, pp. 6108–6113. 41

[92] PEQUITO, S. D., KAR, S., AGUIAR, A. P., ET AL. A framework for structural input/output and control configuration selection in large-scale systems. *IEEE Trans. Automat. Contr. 61*, 2 (2016), 303–318. 33, 41

[93] POLJAK, S. On the generic dimension of controllable subspaces. *IEEE Transactions on Automatic Control 35*, 3 (1990), 367–369. 48

[94] PU, C.-L., PEI, W.-J., AND MICHAELSON, A. Robustness analysis of network controllability. *Physica A: Statistical Mechanics and its Applications 391*, 18 (2012), 4420–4425. 8, 10, 38, 42, 43, 47

[95] RABIN, M. O., AND VAZIRANI, V. V. Maximum matchings in general graphs through randomization. *Journal of Algorithms 10*, 4 (1989), 557–567. 46

[96] RASHID, N., WAN, J., QUIROS, G., CANEDO, A., AND AL FARUQUE, M. A. Modeling and simulation of cyberattacks for resilient cyber-physical systems. In *Automation Science and Engineering (CASE), 2017 13th IEEE Conference on* (2017), IEEE, pp. 988–993. 7, 63

[97] RÉGIN, J.-C. A filtering algorithm for constraints of difference in csps. In *AAAI* (1994), vol. 94, pp. 362–367. 11, 17, 45, 91, 100

[98] RINALDI, S. M., PEERENBOOM, J. P., AND KELLY, T. K. Identifying, understanding, and analyzing critical infrastructure interdependencies. *Control Systems, IEEE 21*, 6 (2001), 11–25. 7

[99] RUGH, W. J., AND RUGH, W. J. *Linear system theory*, vol. 2. prentice hall Upper Saddle River, NJ, 1996. vii, 4, 19

[100] RUTHS, J., AND RUTHS, D. Robustness of network controllability under edge removal. In *Complex Networks IV*. Springer, 2013, pp. 185–193. 9, 10, 43, 53

[101] RUTHS, J., AND RUTHS, D. Control profiles of complex networks. *Science 343*, 6177 (2014), 1373–1376. 47

[102] SANKOWSKI, P. Faster dynamic matchings and vertex connectivity. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), Society for Industrial and Applied Mathematics, pp. 118–126. 41

[103] SÉLLEY, F., BESENYEI, Á., KISS, I. Z., AND SIMON, P. L. Dynamic control of modern, network-based epidemic models. *SIAM Journal on applied dynamical systems 14*, 1 (2015), 168–187. 8, 38

[104] SHIELDS, R., AND PEARSON, J. Structural controllability of multiinput linear systems. *IEEE Transactions on Automatic control 21*, 2 (1976), 203–212. 22, 38

[105] SLAY, J., AND MILLER, M. Lessons learned from the maroochy water breach. In *International Conference on Critical Infrastructure Protection* (2007), Springer, pp. 73–82. 7

[106] SLOTINE, J.-J. E., LI, W., ET AL. *Applied nonlinear control*, vol. 199. Prentice hall Englewood Cliffs, NJ, 1991. 5

[107] SOLOMON, B., IONESCU, D., LITOIU, M., AND ISZLAI, G. Autonomic computing control of composed web services. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (2010), ACM, pp. 94–103. 4

[108] STOUFFER, K., FALCO, J., AND SCARFONE, K. Guide to industrial control systems (ics) security. *NIST special publication 800*, 82 (2011), 16–16. 6, 7

[109] SUN, P. G., AND MA, X. Controllability and observability of cascading failure networks. *Journal of Statistical Mechanics: Theory and Experiment 2017*, 4 (2017), 043404. 10, 43

[110] SUN, S., MA, Y., WU, Y., WANG, L., AND XIA, C. Towards structural controllability of local-world networks. *Physics Letters A 380*, 22 (2016), 1912–1917. 4

[111] TARJAN, R. Depth-first search and linear graph algorithms. *SIAM journal on computing 1*, 2 (1972), 146–160. 28, 40, 59, 86, 98, 128, 131

[112] TASSA, T. Finding all maximally-matchable edges in a bipartite graph. *Theoretical Computer Science 423* (2012), 50–58. 17, 46, 92, 101

[113] VINAYAGAM, A., GIBSON, T. E., LEE, H.-J., YILMAZEL, B., ROESEL, C., HU, Y., KWON, Y., SHARMA, A., LIU, Y.-Y., PERRIMON, N., ET AL. Controllability analysis of the directed human protein interaction network identifies disease genes and drug targets. *Proceedings of the National Academy of Sciences 113*, 18 (2016), 4976–4981. 12, 48, 49, 115

[114] WANG, B., GAO, L., AND GAO, Y. Control range: a controllability-based index for node significance in directed networks. *Journal of Statistical Mechanics: Theory and Experiment 2012*, 04 (2012), P04011. 48

[115] WANG, B., GAO, L., GAO, Y., AND DENG, Y. Maintain the structural controllability under malicious attacks on directed networks. *EPL (Europhysics Letters) 101*, 5 (2013), 58003. 9, 42, 53

[116] WANG, H. Robustness of networks. 42

[117] WANG, L.-Z., CHEN, Y.-Z., WANG, W.-X., AND LAI, Y.-C. Physical controllability of complex networks. *Scientific reports 7* (2017), 40198. 8, 38

[118] WANG, W.-X., NI, X., LAI, Y.-C., AND GREBOGI, C. Optimizing controllability of complex networks by minimum structural perturbations. *Physical Review E 85*, 2 (2012), 026115. 9, 53

[119] WATTS, D. J., AND STROGATZ, S. H. Collective dynamics of 'small-world'networks. *nature 393*, 6684 (1998), 440–442. 8, 37, 43

[120] XIAO, Y.-D., LAO, S.-Y., HOU, L.-L., AND BAI, L. Edge orientation for optimizing controllability of complex networks. *Physical Review E 90*, 4 (2014), 042804. 53

[121] YAN-DONG, X., SONG-YANG, L., LV-LIN, H., AND LIANG, B. Optimization of robustness of network controllability against malicious attacks. *Chinese Physics B 23*, 11 (2014), 118902. 42, 53

[122] YANG, S.-H., SMOLA, A. J., LONG, B., ZHA, H., AND CHANG, Y. Friend or frenemy?: predicting signed ties in social networks. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (2012), ACM, pp. 555–564. 45

[123] ZDEBOROVÁ, L., AND MÉZARD, M. The number of matchings in random graphs. *Journal of Statistical Mechanics: Theory and Experiment 2006*, 05 (2006), P05003. 48

[124] ZHANG, P. *Handbook of graph theory*. Chapman and Hall/CRC, 2013. 41

[125] ZHANG, S., AND WOLTHUSEN, S. Driver-node based security analysis for network controllability. pp. 1–6. 17th European Control Conference (ECC19), ECC 2019 ; Conference date: 25-06-2019 Through 29-06-2019. doi:10.23919/ECC.2019.8796264. i, 14

[126] ZHANG, S., AND WOLTHUSEN, S. Structural controllability recovery via the minimum-edge addition. pp. 1–6. 2019 AMERICAN CONTROL CONFERENCE ; Conference date: 10-07-2019 Through 12-07-2019. i, 13

[127] ZHANG, S., AND WOLTHUSEN, S. D. Iterative recovery of controllability via maximum matching. In *Automation Science and Engineering (CASE), 2017 13th IEEE Conference on* (2017), IEEE, pp. 328–333. doi:10.1109/COASE.2017.8256124. i, 13

[128] ZHANG, S., AND WOLTHUSEN, S. D. Efficient analysis to protect control into critical infrastructures. In *International Conference on Critical Information Infrastructures Security* (2018), Springer, pp. 226–229. doi:10.1007/978-3-030-05849-4_18. i, 13, 14

[129] ZHANG, S., AND WOLTHUSEN, S. D. Efficient control recovery for resilient control systems. In *Networking, Sensing and Control (ICNSC), 2018 IEEE 15th International Conference on* (2018), IEEE, pp. 1–6. doi:10.1109/ICNSC.2018.8361318. i

[130] ZHANG, S., AND WOLTHUSEN, S. D. Security-aware network analysis for network controllability. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)* (2018), IEEE. doi:10.1109/WAINA.2018.00136. i, 13

[131] ZHANG, X., HAN, J., AND ZHANG, W. An efficient algorithm for finding all possible input nodes for controlling complex networks. *Scientific Reports 7*, 1 (2017), 10677. 47